UNIVERSIDAD POLITECNICA DE MADRID

ESCUELA TECNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACION

TRABAJO FIN DE GRADO
2014

Desarrollo de un sistema portátil de monitorización personal

Alumno: Sergio Ruiz García
Tutor: Pedro Guerra Gutiérrez
Ponente: Georgios Kontaxakis Antoniadis

Miembros del Tribunal: Georgios Kontaxakis
María Jesús Ledesma Carbayo
Fernando González Sanz
Andrés de Santos Lleó

Fecha de Lectura y Defensa:

Calificación:

UNIVERSIDAD POLITECNICA DE MADRID

ESCUELA TECNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACION

TRABAJO FIN DE GRADO
2014

Desarrollo de un sistema portátil de monitorización personal

Alumno: Sergio Ruiz García
Tutor: Pedro Guerra Gutiérrez
Ponente: Georgios Kontaxakis Antoniadis

Miembros del Tribunal: Georgios Kontaxakis
María Jesús Ledesma Carbayo
Fernando González Sanz
Andrés de Santos Lleó

Fecha de Lectura y Defensa:

Calificación:

Resumen

Un sistema de monitorización personal está pensado para mantener un control constante de ciertos parámetros vitales, de forma que se pueda realizar un registro de los mismos o generar algún tipo de alarma si se salen fuera de sus parámetros habituales o alcanzan cotas de riesgo. En este aspecto, se convierten en una opción cada vez más atractiva cuanto menos invasivos son, de forma que el objetivo es conseguir un sistema que monitorice al paciente sin entorpecer sus acciones cotidianas. Por este motivo, los dispositivos *wearables* son una buena opción. Un reloj, un colgante o una pulsera son elementos que llevan muchas personas, y por tanto, susceptibles de incorporar un procesador y algunos sensores que realicen las medidas.

En este Trabajo de Fin de Grado se pretende realizar un prototipo sencillo de un sistema de monitorización personal que ilustre el funcionamiento de una red de área personal (WBAN) a partir de una plataforma de desarrollo preexistente. La plataforma en cuestión es el eZ430-Chronos de Texas Instruments, un *System on Chip* que incorpora sensores de aceleración, temperatura y presión. El *System on Chip* se encapsula en la forma de un reloj de pulsera. Además, se dispone de una banda, fabricada por BM innovations, que permite medir el ritmo cardíaco.

En primer lugar se hará un análisis del sistema disponible, por un lado de la arquitectura hardware y firmware del dispositivo, y por otro lado de la arquitectura del software del cliente para PC. El firmware disponible en un principio permite únicamente la captura y registro de algunos parámetros del entorno, así como de las pulsaciones. Adicionalmente, el eZ430-Chronos dispone de un cliente para PC que le permite descargar los datos almacenados en la memoria flash al PC, así como configurar ciertos valores.

En una segunda fase, se modificará el firmware para convertirlo en un sistema de monitorización personal, en el que se le retira al usuario la capacidad de control sobre la ejecución y se automatizan los procesos de adquisición y descarga de datos. Además, se creará una aplicación para PC que tenga la misma funcionalidad que el software original, aparte de incluir algunas características adicionales.

Palabras clave: WBAN, red de área personal, monitorización, e-health, eZ430-Chronos, IEEE 802.15.6, caídas.

Summary

A personal monitoring system is intended to keep a constant control over certain vital parameters, so that a record of them can be kept, or some alarm can be generated if they are out of their usual parameters or reach a life-threatening level. In this aspect, the less invasive they are, the better option they become. Therefore, the objective is to obtain a monitoring system that can measure vital parameters without impeding the patient's daily actions. For this reason, wearable devices are a good option, as lots of people wear watches, pendants or bracelets and are thus susceptible to incorporate a processor and some sensors to obtain the measures.

With this undergraduate thesis, the building of a simple prototype of a personal monitoring system illustrating the operation of a Wireless Body Area Network (WBAN) from an existent developing platform is intended. Said platform is Texas Instrument's eZ430-Chronos, a System on Chip incorporating acceleration, temperature and pressure sensors that comes as a wristwatch. Moreover, a chest strap commercialised by BM innovations is available. The strap allows for a heartbeat sensing.

In the first place, and analysis of the available system needs to be made - on one hand of the hardware and firmware architecture of the device, and on the other hand, of the architecture of the software of the PC application. The firmware available allows just for the capture and recording of some environmental parameters, as well as the heartbeating. Additionally, the eZ430-Chronos has a PC client that provides with the ability to download the data stored in the flash memory to the PC, as well as configuring some internal values.

On a second stage, the firmware will be modified to transform it into a personal monitoring system, in which the user's ability to control the execution flow is denied, and the acquisition and syncing process are automated. Moreover, an application for PC with the same functionality as the original software will be created, besides including some additional features.

Keywords: WBAN, Body Area Network, monitoring, e-health, eZ430-Chronos, IEEE 802.15.6, falls

Índice

Índice de figuras

Introducción

En los últimos tiempos se ha popularizado el concepto de red de área corporal (en inglés [*Wireless*] *Body Area Network* WBAN o BAN), que describe una red de comunicación inalámbrica entre dispositivos de baja potencia situados en el cuerpo (tanto dentro como fuera de éste) de manera no intrusiva. Dicha red se puede utilizar para la comunicación de sensores y actuadores con dispositivos que se encargan de su procesado y almacenamiento. En el caso de los sensores es habitual la adquisición tanto de parámetros vitales, movimientos, como otros parámetros del entorno.

De esta manera resulta posible la monitorización remota de pacientes o personas necesitadas de un control permanente, ya que los dispositivos de procesado y almacenamiento ofrecen también la posibilidad de conectarse a un servidor en el que descargan la información. Esta información se introduce en una base de datos lista para que la consulte un profesional que se puede encontrar en cualquier punto del globo, o se procesa en línea con el fin de detectar situaciones en las que la vida del paciente pueda estar en riesgo (por una parada cardíaca, por ejemplo) y emprender alguna acción, como avisar al servicio de emergencias.

Este tipo de soluciones de telemedicina incrementa la autonomía del paciente, que ya no necesita acudir a una consulta médica para cierto tipo de pruebas o medidas, así como la libertad de desplazamiento que ofrece el tener un sistema portátil no invasivo. Otra de las ventajas es el ahorro de tiempo y de recursos humanos; una parte importante del tiempo de las consultas médicas se consume en la adquisición de datos que, de llevar implementado un sistema WBAN, estarían ya disponibles para su uso. Las consultas médicas se agilizarían y sería posible tratar a más pacientes en una jornada. Además, con la adquisición masiva de datos de distintos pacientes se podría recurrir a técnicas de *Big Data* para contrastar datos y ayudar al profesional a realizar un diagnóstico más rápido.

Respecto de los actuadores, también proporcionan libertad al paciente en el sentido de que si sufre una enfermedad crónica no necesita estar pendiente de tomar la medicación. El sistema sabe cuándo debe administrar la dosis, la cantidad de dosis a suministrar en caso de que la dosis varíe bajo ciertas condiciones (insulina en el caso de pacientes diabéticos, por ejemplo), y cuándo la reserva de medicamento se va a terminar. De esta manera el paciente ya no tiene que preocuparse por llevar sus medicinas cuando se desplace de su vivienda.

Objetivos

Los objetivos a conseguir con la realización del presente Trabajo de Fin de Grado son los siguientes:

- **Realización de un prototipo de un sistema de monitorización.** Se pretende realizar un sencillo sistema de monitorización en el que usuario cuenta con una pulsera que gestiona la WBAN y envía los datos a un equipo externo. Para explorar las posibilidades se decidió trabajar sobre un kit de desarrollo existente, consistente en un reloj de pulsera programable y desarrollar una demo aprovechando algunas de las características hardware y software del dispositivo, así como los ejemplos que ya vienen con el kit. De este modo se persigue que con algunas modificaciones incorporadas el prototipo, pudiera constituir la base de un producto comercial. De igual manera, la extensión permitirá obtener un sistema WBAN sencillo a partir de una plataforma que vuelve al apartado docente: cómo obtener la aplicación deseada a partir de un sistema preexistente.
- **Análisis y comprensión de la plataforma eZ430-Chronos.** De cara a realizar posibles proyectos futuros sobre la plataforma, o la utilización de la misma para docencia, era necesario un análisis del firmware y el hardware disponibles, para ordenar los ejemplos existentes y tener una solución propia y documentada. Por este motivo uno de los objetivos adicionales del proyecto era la comprensión de los recursos proporcionados, así como hasta qué punto se pueden modificar para realizar nuestras aplicaciones y la potencia de los recursos existentes. En el aspecto docente es especialmente importante, ya que el sistema ofrece un ejemplo sencillo de aplicaciones que se pueden conseguir, pero con unas demos muy cerradas que impedirían a futuros estudiantes su utilización.
- **Análisis y realización del cliente para PC.** Teniendo en cuenta que se puede desear realizar modificaciones en el firmware del eZ430-Chronos, una opción interesante es permitir la comunicación con el PC, una característica que viene incluida con el firmware existente con una aplicación relativamente cerrada. Por tanto, es pertinente también un análisis del protocolo de comunicaciones entre el dispositivo y el PC, a través del punto de acceso disponible, así como la clonación del cliente para PC mediante el uso de un lenguaje de programación que permita la multiplataforma, tales como Java o Python.

Estado del arte

Las WBAN son un concepto relativamente nuevo. A pesar de que los sistemas de control del entorno para personas con una grave discapacidad no son nuevos y están disponibles desde los años 60, son caros y están muy especializados, proporcionando un número reducido de funciones[11]. Las WBAN aplicables masivamente en pacientes tienen, por tanto, un gran atractivo para la sociedad. Además, las personas mayores representan un grupo de alto riesgo en lo que respecta a los accidentes domésticos. Las caídas en un entorno doméstico son ejemplos recurrentes de este tipo de accidentes, tras las que puede seguir un periodo largo de hospitalización[11]. Este tipo de redes requieren dispositivos con unas características muy específicas, que presentan algunos problemas.

Entre los problemas que se presentan se encuentra la vida de la batería. Los dispositivos deben ser de bajo consumo, ya que en algunos casos los sensores se colocan en el interior del cuerpo y reemplazar las baterías normalmente conlleva una operación. En este aspecto, son deseables los dispositivos con menor consumo. Algunos dispositivos incorporan técnicas de *energy harvesting*, mientras que otros se alimentan a través de inducción RF**[19]**.

Otro de los problemas es el tamaño. Los sensores y el dispositivo que actúe como acumulador deben ser suficientemente pequeños, así como tener una forma ergonómica para no entorpecer la rutina de los usuarios. Un dispositivo que sea demasiado grande o pesado será visto como un estorbo y no será bien acogido por los pacientes. Además, en el caso de los sensores colocados en el interior del cuerpo, debe ser lo menos invasivos posible para que no produzcan sensaciones incómodas a los usuarios.

Respecto de la gestión de datos, supone un problema de infraestructura. Si se recogen datos de gran cantidad de pacientes, la cantidad de datos que se maneje será ingente, dando lugar a la necesidad de grandes bases de datos para conseguir su persistencia. De igual manera, se necesitará una infraestructura de comunicación que conecte los acumuladores con estas bases de datos. También hay que garantizar la consistencia de datos, para evitar que datos obtenidos en distintos lugares no se comuniquen a los demás y distintos agentes manejen informaciones diferentes.

Por último, otro aspecto crítico es la interoperabilidad. Al no estar aún muy extendido, no todos los sistemas existentes son compatibles entre ellos. En este aspecto, los distintos sistemas que se implementen deben ser capaces de comunicarse entre sí, de forma que cada usuario pueda ser atendido por cualquier sistema WBAN implementado, independientemente de cuáles dos sean estos sistemas utilizados.

Como se ha comentado, las WBAN han sido objeto de investigaciones. Entre alguno de los sistemas desarrollados se encuentra el realizado por varios investigadores del IEEE[11], un sistema equipado con un método de vigilancia autónomo integrado con el entorno, que proporciona monitorización constante de las actividades y parámetros fisiológicos de las personas mayores. Este sistema reconoce cambios en el estado de salud, y tiene la capacidad de alertar a doctores o familiares. El sistema se basa en un conjunto de sensores que envían información sobre la salud y el entorno en tiempo real, y estos datos se envían a actuadores en el hogar o personales. También permite avisar a

las personas a cargo de su cuidado, a familiares o a doctores si se detecta un estado de alarma, y una vez proporcionada la asistencia, los doctores pueden introducir manualmente información que permita mejorar o actualizar el tratamiento o el seguimiento del paciente. Desde el punto de vista del usuario, le permite ser algo más independiente de los doctores o las personas que les cuidan, pudiendo éstos ser alertados únicamente cuando hay una alerta, además de beneficiarse de los actuadores, que pueden realizar tareas sencillas como regular la temperatura o suministrarles la dosis necesaria de medicamento sin la necesidad de que haya otra persona haciéndolo.

Otro sistema existente[12] permite monitorizar los niveles de actividad de las personas mayores mediante dos acelerómetros. Se colocan dos acelerómetros, uno en el centro del pecho y otro en el muslo, y pueden medir a través de la aceleración estática (la gravedad) y a través de la aceleración dinámica si el sujeto está sentado, de pie, tumbado o en movimiento. Los datos de los acelerómetros se leen y se comparan con umbrales predefinidos, para posteriormente ser cifrados y enviados por SMS a un servidor remoto.

Hay otros sistemas capaces de detectar cuándo el paciente sufre una caída. En el caso de un sistema desarrollado en la universidad de Sevilla[13], se colocan en el centro de gravedad dos acelerómetros que miden cuatro ejes, de forma que con tres de ellos se puede monitorizar el movimiento y analizar la forma de caminar de la persona. De igual manera sirve para investigar las causas de las caídas en caso de producirse, así como modificaciones en la actividad diaria. Se hace uso de los acelerómetros a través de un parche pegado a la piel situado a la altura del hueso sacro. Uno de los requisitos era que se monitorizara durante las 24 horas, de forma que utilizaron una WPAN para independizar el sensor de otros elementos periféricos, como por ejemplo el botón de emergencia. Los datos recogidos se envían a un servidor de manera inalámbrica a través de la red telefónica en caso de encontrarse en el hogar, o con un cliente WPAN adicional mediante una conexión serie a un teléfono móvil.
Otro sistema posterior[14] permite detectar caídas en el hogar, el hospital, la residencia, o cualquier lugar cerrado que incorpore dicho sistema, a partir de las vibraciones detectadas en el suelo y del sonido que la caída produce. Las señales de las vibraciones recibidas se procesan y se comparan con señales de referencia con las que se ha entrenado previamente al sistema. Para mitigar el efecto del ruido, se activa el algoritmo de clasificación de vibraciones únicamente cuando la señal recibida tiene un nivel significativo. Una vez que se detecta el evento de vibración, se analiza la señal acústica, para distinguir entre una caída de una persona y cualquier otro evento que produzca una vibración similar.
Respecto del problema comentado de interoperabilidad, la organización IEEE creó en 2007 un subgrupo de trabajo dentro del grupo 802.15, el 806.15.6, cuyo objetivo era emitir un estándar para las WBAN. En 2007 se emitió la versión definitiva del estándar, en el que se definen las especificaciones de las capas MAC y PHY, de forma que se unificaran los formatos de trama y las modulaciones utilizadas.

Las especificaciones de la capa MAC establecen que se pueden trabajar en varias bandas: NB (*Narrow Band*), UWB (*Ultra-Wide Band*) y HBC (*Human Body Communications*). Se define una topología en estrella con uno o dos saltos, y un único *hub*. El periodo de transmisión se puede dividir en varias tramas de igual duración, que permiten establecer distintos modos de operación; por ejemplo, si hace falta cursar tráfico de alta prioridad o emergencia hay tramas con prioridades, y si es necesario se

descartará tráfico no prioritario para asegurar la transmisión del prioritario. También ofrece la posibilidad de trabajar sin asignaciones temporales, mediante un método de peticiones, en el que un nodo realiza una petición cuando desea transmitir, y empieza cuando el *hub* le da permiso.

Para la capa PHY se contempla el uso de dos bandas obligatorias (HBC y UWB) y una opcional (NB). El nivel PHY en 802.15.6 se encarga de la activación y desactivación del transceptor radio, la evaluación de la ocupación del canal y de la transmisión y recepción de datos. Proporciona mecanismos sencillos de detección y corrección de errores, tales como CRC y códigos BCH.

Arquitectura de la solución propuesta

Un sistema de monitorización personal pretende tener controlados ciertos parámetros del paciente de forma continuada. Por ello, es necesario realizar medidas de manera periódica para registrar los valores que proporcionan los sensores. Además, al tratarse de un sistema portátil, existen importantes limitaciones de consumo de batería y de espacio de almacenamiento que obligan a comunicarse con un dispositivo externo, por ejemplo un PC o un *smartphone*, en el que se descargarán los datos para garantizar su persistencia o ser tratados

Por tanto, la solución propuesta consiste en un conjunto de sensores que se comunican con un procesador que actúa como *hub* (concentrador) para transmitir el valor medido por cada uno de los sensores. Dicho procesador se encarga de gestionar el encendido y apagado de los módulos, así como su funcionamiento. Cuando reciba los datos de los sensores debe almacenarlos en la memoria, y cuando no sea posible seguir introduciendo nuevos datos, se avisará al punto de acceso conectado al PC para iniciar una descarga de la memoria al completo.

Por su parte, el programa de PC debe proporcionar la capacidad de controlar el reloj a través del punto de acceso, ya que en la comunicación el PC es el maestro y el reloj es el esclavo. Desde el cliente para PC el encargado de configurar el reloj debe ser capaz de cambiar los valores de los parámetros que se miden y con qué intervalo se hace, así como otros cambios adicionales y no relacionados con la monitorización, como la hora o la fecha. Cuando se conecte con el reloj y se descargue la memoria flash del dispositivo, debe almacenarla en un archivo de tal forma que se pueda crear un repositorio con un histórico de medidas, y proporcionar la posibilidad de hacer algún tipo de procesado de los datos.

De tal forma que se obtiene la siguiente arquitectura:

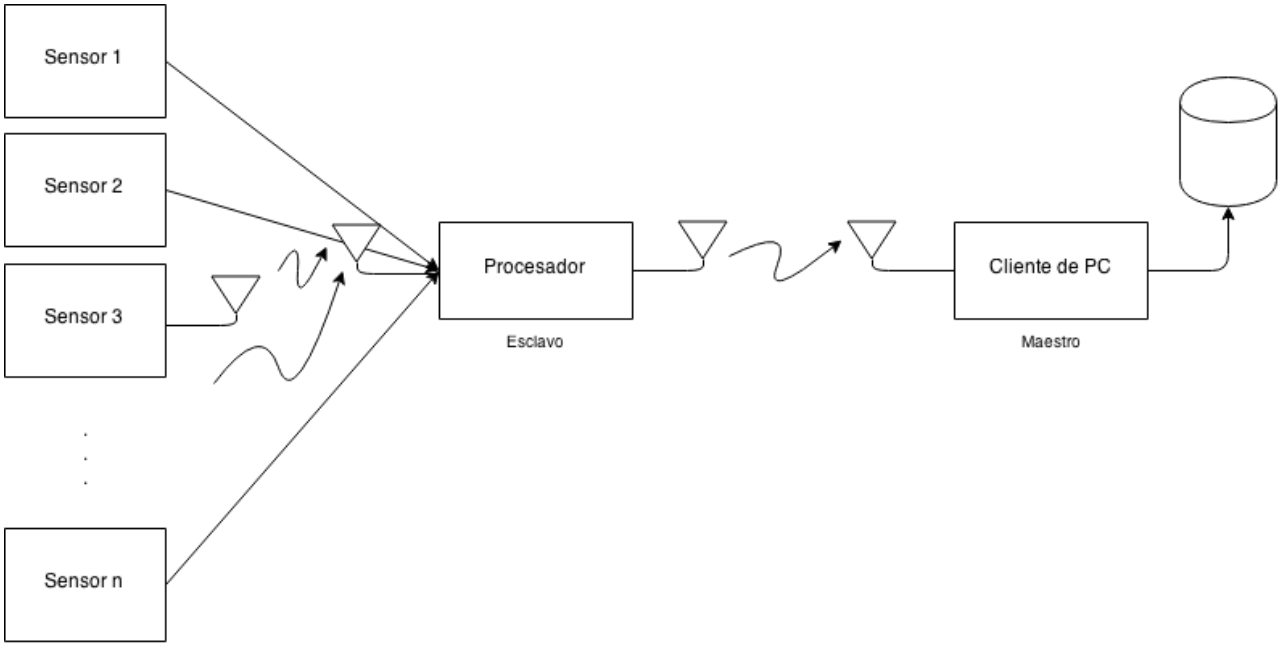

Figura 1 - Arquitectura de la solución

Material y Métodos

Material. El prototipo de desarrollo eZ430-Chronos

Dentro de la serie eZ430-Chronos hay dos productos, el eZ430-Chronos Black Board y el eZ430-Chronos White Board. La diferencia notable, dejando de lado el detalle del color de la PCB del *Access Point*, reside en el sensor de presión que montan uno y otro. Por lo demás, son idénticos. En el presente TFG nos centraremos en la versión disponible: Black Board. En la siguiente figura se muestra un diagrama con los módulos disponibles[4].

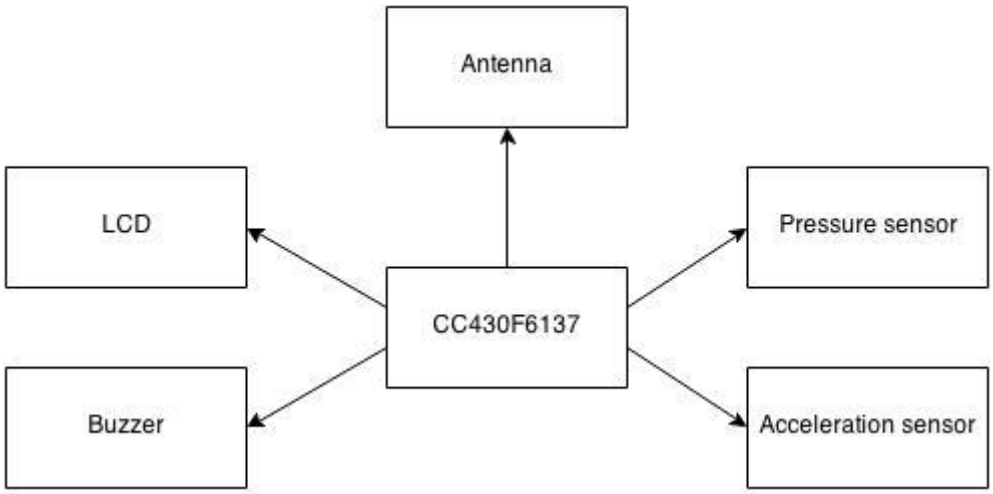

Figura 2 - Diagrama con los módulos disponibles

El reloj incorpora un microprocesador (MCU) de Texas Instruments perteneciente a la familia CC430, el CC430F6137[5]. Dicha familia está compuesta por microcontroladores de ultra bajo consumo con transceptores RF integrados[8] y un MCU con la misma arquitectura que los MSP430. Son microcontroladores especialmente diseñados para aplicaciones portables, con un consumo muy bajo y modos de ahorro de batería (*Low Power Modes*).
Se dispone además de un acelerómetro, el CMA3000-D01[5] de VTI, que proporciona medidas de aceleración en tres ejes. Está pensado especialmente para fabricación en masa en aplicaciones de bajo consumo y precio, y espacio reducido[6]. En su configuración inicial genera una interrupción por el pin INT cada vez que tiene una medida nueva.

Para medir la altura, la presión, y la temperatura se dispone del sensor de presión SCP1000-D11[5], también de VTI. Es un sensor de presión absoluta con interfaz I2C[7], aunque también admite una interfaz SPI. Además, posee también un sensor de temperatura, lo cual nos permite usar el chip para varias aplicaciones. Para realizar las medidas no necesitamos más que acceder a través de la interfaz serie.

El prototipo incorpora un display LCD no comercial, fabricado especialmente para la plataforma. También se incorpora una pequeña antena sintonizada, en el caso de los kits distribuidos en Europa, a 868 MHz. Esta frecuencia queda dentro de la banda WMTS (*Wireless Medical Telemmetry Services*) definida por el FCC americano. En su definición original no se encuentra en esta banda, pero en Europa se debatió y se decidió no usar la establecida en EEUU por estar ya ocupada[16], por lo que la banda utilizada en Europa para WMTS es 836 - 870 MHz[3].

Las conexiones de los elementos se describen en la siguiente figura[9]:

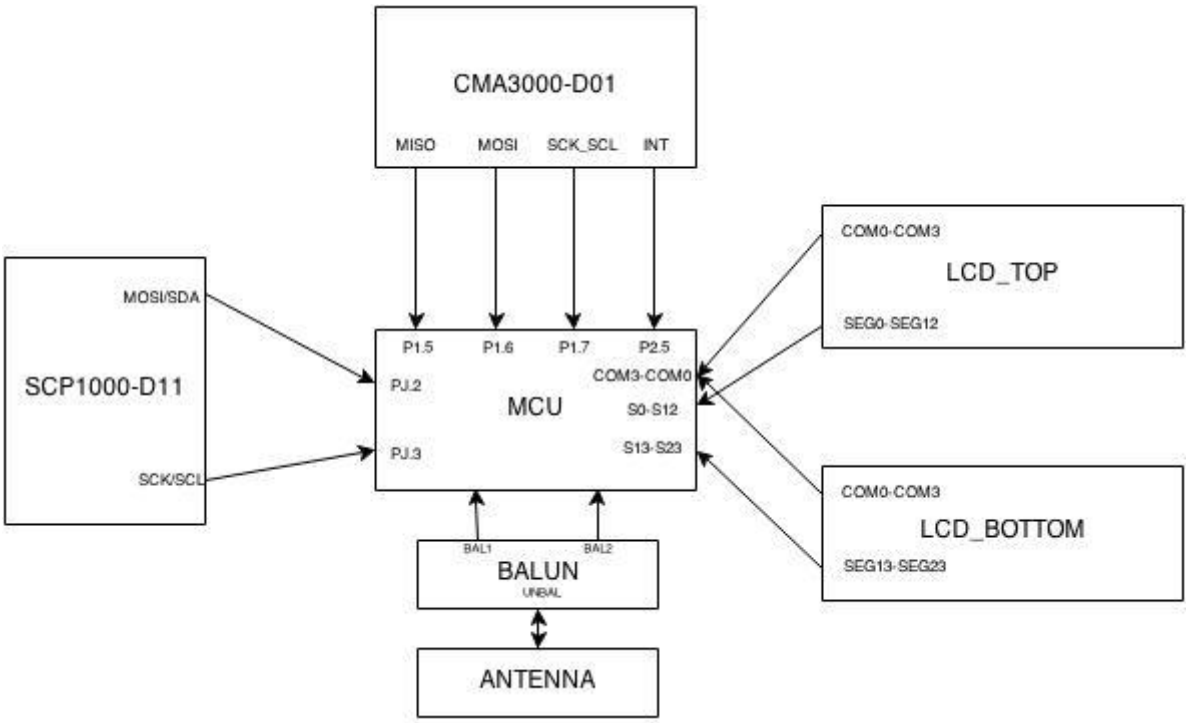

Figura 3 - Conexiones de los elementos

La antena se comunica con el punto de acceso a una frecuencia de 868 MHz, mediante un protocolo de TI llamado SimpliciTI. El punto de acceso es un *dongle* USB[10] compuesto por un CC1111F32RSP que actúa de controlador, una antena con un *balun* para comunicarse con el reloj, y una interfaz USB para la conexión con el PC. Cuenta además con un reloj interno propio.

Material. Banda para medir el ritmo cardíaco del usuario

La antena del eZ430-Chronos permite la comunicación tanto con el punto de acceso externo como con otros dispositivos. En este caso, permite comunicarse con una banda de pecho para medir el ritmo cardíaco del usuario.

Dicha banda está fabricada por BMi, modelo BM-CS5EU. Entre ellas utilizan un protocolo propio creado por IAR Systems llamado BlueRobin, que proporciona una transmisión de datos de baja potencia[17][18]. Cada banda tiene un ID de 24 bits propio que permite identificarlo unívocamente y conectar la banda con un receptor. La prevención de colisiones permite que hasta 200 bandas operen en el rango de un mismo receptor. En Europa opera en la banda de frecuencia de 868 MHz, que corresponde como se ha comentado anteriormente a la banda WMTS. La alimentación se realiza mediante una pila estándar CR2032.

Arquitectura de un microcontrolador CC430

La arquitectura del microcontrolador CC430F6137 que controla el sistema se distingue de otros modelos de la familia MSP430 de Texas Instruments en que integra un módulo de RF y un controlador para el LCD. Dicha arquitectura se representa en la siguiente figura[8]:

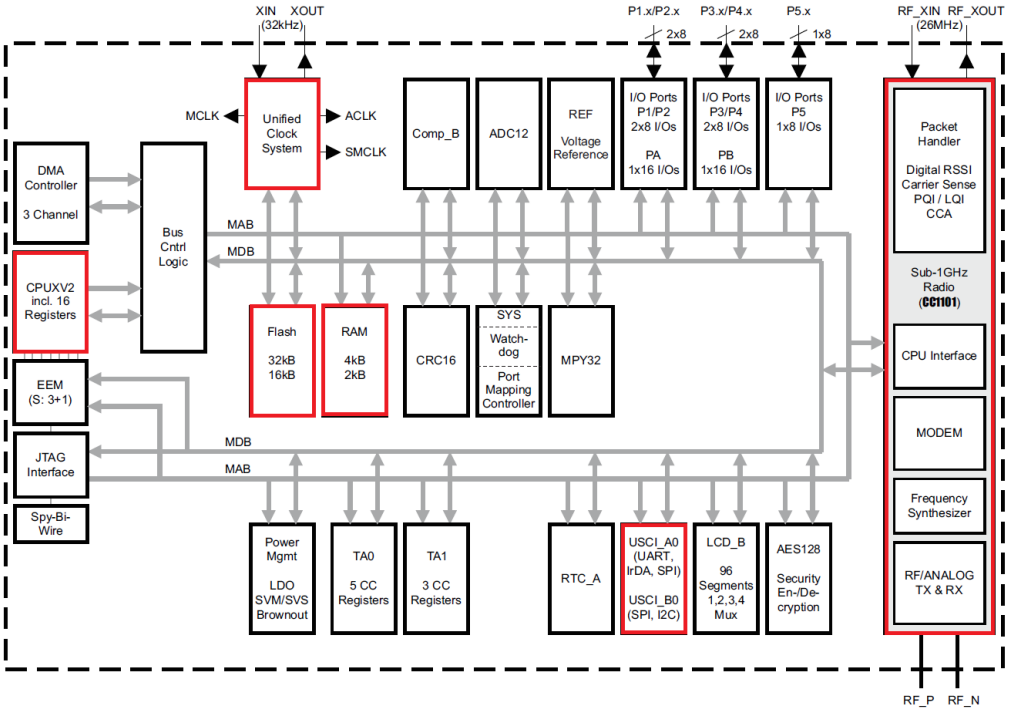

Figura 4 - Arquitectura de un CC430

En rojo se marcan los módulos más importantes, que se comentan a continuación:

- La CPU tiene una arquitectura RISC de 16 bits. Posee 16 registros que ofrecen un tiempo de ejecución reducido; el tiempo de ejecución de operaciones entre registros es un ciclo de reloj. Las instrucciones pueden operar sobre datos de tamaño *word* y *byte*. Posee seis modos de operación, uno activo y cinco de bajo consumo, que se comentan un poco más abajo.
- La memoria flash puede ser programada a través de la interfaz con el PC o durante la ejecución del programa. Se pueden escribir *byte*, *words* y *longs* a la memoria. La memoria se divide en páginas de 512 *bytes*, que se reparten en cuatro segmentos de información de 128 *bytes* cada uno. Cada página se puede borrar individualmente si es necesario. Los segmentos de información también se pueden borrar individualmente. En el caso del CC430F6137 la memoria flash tiene un tamaño de 32kB.
- La memoria RAM se divide en n sectores de 2kB cada uno. Cada sector se puede apagar completamente para ahorrar batería, sin embargo, se pierde la capacidad de retención de datos. Cada sector entra automáticamente en el modo de ahorro cuando es posible. La memoria RAM disponible tiene un tamaño de 4kB.
- El módulo USCI se usa para comunicación serie. Proporciona conectividad con protocolos síncronos como SPI e I^2C, y con protocolos asíncronos como UART.
- El módulo radio sub-1GHz está basado en el chip CC1101, que requiere muy pocos componentes externos. La radio utiliza un receptor de frecuencia intermedia baja (low-IF receiver). Para transmisión se utiliza síntesis directa de frecuencia, y se trabaja en varias bandas. Posee una memoria a la que puede acceder la CPU para consultar el estatus, comprobar la configuración, y acceder a los datos. No tiene capas PHY o MAC formales, así que es la radio la que se encarga de enviar y recibir directamente los datos[15].

Mención especial merecen los modos de bajo consumo LPM0-LPM4. En dichos modos se van apagando módulos del sistema (CPU, algunos relojes, etc) de forma que se pueda reducir la corriente de drenado del dispositivo y de esta manera poder alargar la vida de la batería. En el modo LPM0 es en el que menos módulos se apagan, siendo éstos únicamente la CPU y el reloj de sistema. En el modo de mínimo consumo (LPM4) se apagan todos los modos y lo único que queda activo son las persistencias de datos y las interrupciones, que es la única manera de salir de este modo.

Con estos modos se puede alargar bastante la vida de la batería, ya que se consiguen consumos muy bajos, como es por ejemplo el del modo LPM3, el que se usa en la aplicación del reloj, que tiene un valor medio de 8.9 μA. Si únicamente se utiliza la función de hora y fecha, el consumo sube hasta 9 μA, permitiendo una duración de la batería de hasta 27.7 meses[7]. Para el sistema se utiliza el modo LPM3 porque debe actuar también como reloj aparte de como WBAN. No es posible utilizar el modo de más bajo consumo, ya que en él ningún temporizador se encuentra activo, y no habría forma de medir cuál es el tiempo que pasa entre una interrupción y otra (ya que para poder salir del modo LPM4 es necesario una interrupción externa).

Software de Desarrollo empleado

Para el desarrollo del software empotrado se emplea la herramienta Code Composer Studio v5.1 de Texas Instruments[1], un entorno de desarrollo basado en Eclipse. El lenguaje en el que se programa el dispositivo es C. La propia herramienta provee un compilador a tal efecto, además de la capacidad de cargar el programa en el dispositivo.

La programación del cliente de PC se ha realizado en el lenguaje de scripting Python, un lenguaje interpretado. Por tanto, no tiene entorno de desarrollo pero es necesario tener instalado el intérprete de comandos. La versión del intérprete que se ha utilizado es Python v3.4.1.

Por último, para el análisis de paquetes enviados a través del puerto USB se ha recurrido a dos programas, USBPcap v1.0.0.7 y Wireshark v1.10.8. Con el primero se monitoriza el puerto y se genera un archivo de salida en formato .out a partir de los datos que pasan por él, y el segundo tiene como objetivo interpretar ese archivo de salida mostrándolos en un formato comprensible.

Implementación

Decisiones de diseño

Como la aplicación deseada originalmente era la de recoger parámetros vitales y transmitirlos al PC, se toma como referencia de partida una demo existente, denominada *DataLogger*. La intención es que se pueda usar en personas mayores o con problemas físicos o mentales, de forma que se plantean varios objetivos dentro de la edición del firmware:

- Uno de los objetivos de la extensión de la demo es que todas las funcionalidades que ofrece la plataforma sean transparentes para el usuario; es decir, que únicamente tenga control sobre la hora y la fecha que se muestran. El resto de modos (adquisición de datos, sincronización con el PC, etc) y la transición entre ellos se debe controlar automáticamente.
- Otro objetivo es implementar un "botón del pánico", para que el usuario pueda solicitar asistencia cuando sienta que la necesita pero el sistema no detecte nada extraño. En este caso se implementará con la pulsación de dos botones simultáneamente, ya que es fácil pulsar accidentalmente cualquiera de ellos. De esta manera, cuando se active la alarma será más probable que no sea una falsa alarma.
- Por último, el sistema debe ser capaz de detectar caídas, y ponerse en contacto con el PC para alertar de ella en caso de una detección. Este objetivo se conseguirá aprovechando los modos de funcionamiento que ofrece el acelerómetro.

1 12500 TI Boulevard, Dallas, Texas

Arquitectura del SW embebido en el ejemplo de partida

Con la plataforma eZ430-Chronos se proporcionan dos programas de ejemplo, que se pueden cargar en el reloj. El primero es uno llamado *SportsWatch*, que permite medir la temperatura, la altura, la aceleración y el ritmo cardíaco, además de tener la opción de programar una alarma o tener un cronómetro; el segundo es el llamado *DataLogger*, que nos permite medir la temperatura, la altura y el ritmo cardíaco y almacenar los datos en la memoria flash. Por este motivo necesita reservar gran parte de la memoria disponible para usarla durante la ejecución, de modo que no hay espacio para tener tantas funcionalidades como el otro programa. Por supuesto, en ambos casos tiene la funcionalidad original del reloj.

Para comprobar cuál es la experiencia de usuario de forma que se pueda replicar en la manera de lo posible tras aplicar los cambios, se carga el programa *DataLogger* y se prueba desde el punto de vista de un usuario. Se obtiene el siguiente diagrama de flujo:

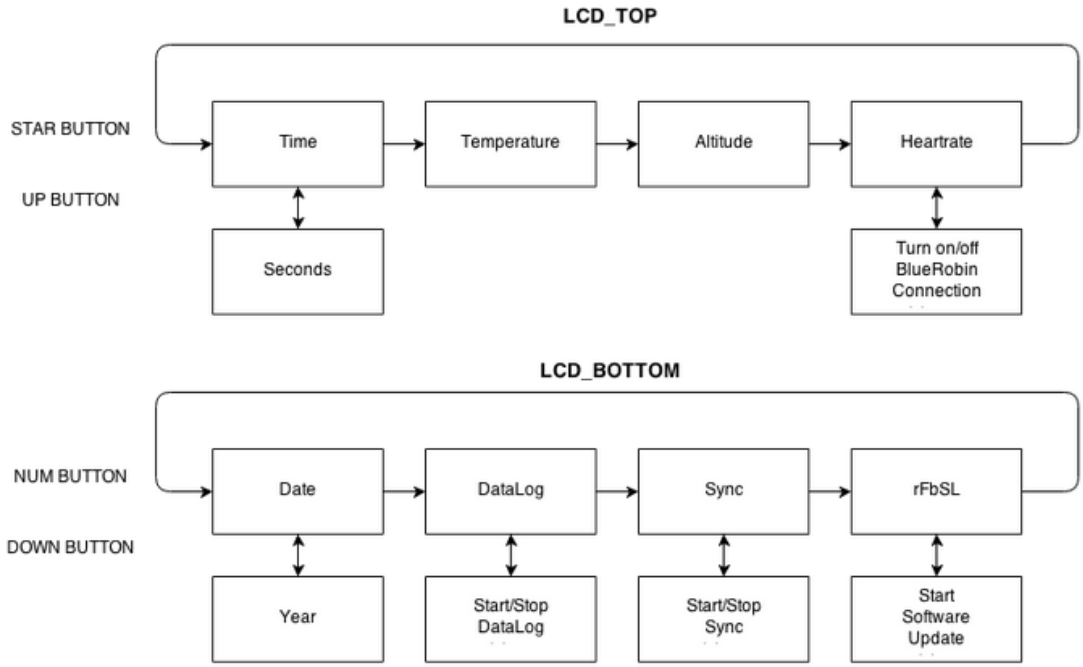

Figura 5 - Diagrama de flujo de ejecución de la estructura de partida

En la pantalla se muestran, al iniciar el programa, la hora y la pantalla del modo *DataLog*.

Para navegar por el menú de la línea superior se utiliza el botón *STAR*. Con cada pulsación se cambia de función, de modo que se puedan ver los distintos datos que muestra la primera línea: hora, temperatura, altitud y ritmo cardíaco. Con el botón *UP* se cambia entre las funciones activables de cada modo, en caso de que la tengan. De este modo, si el modo actual es por ejemplo el de ritmo cardíaco, pulsando el botón *UP* se puede encender o apagar la conexión con la banda para el pecho (función asociada a ese modo).

Para poder navegar por el menú de la línea inferior se ha hecho uso del botón *NUM*, lo que permite visualizar los datos de la línea inferior: fecha, registro de datos, sincronización con el PC y actualización del firmware por RF. Con el botón *DOWN* se accede a las funciones asociadas a ellos, que activan y desactivan las funciones activables de cada modo.

En la siguiente figura se puede ver una imagen frontal del dispositivo, con los botones indicados sobre ella. El botón que aparece en el lado derecho en el centro se utiliza para iluminación, aunque por supuesto se puede reasignar.

Figura 6 - Imagen frontal del eZ430 - Chronos

Para una comprensión más fácil se comentarán los ficheros más importantes y las relaciones que se producen entre ellos. De igual manera, se ilustran los flujos de ejecución y diagramas de estado de algunos de los modos disponibles.

Main

Comenzando con el archivo principal (main.c), que es el que nos define la ejecución del programa, se tiene el siguiente flujo de ejecución:

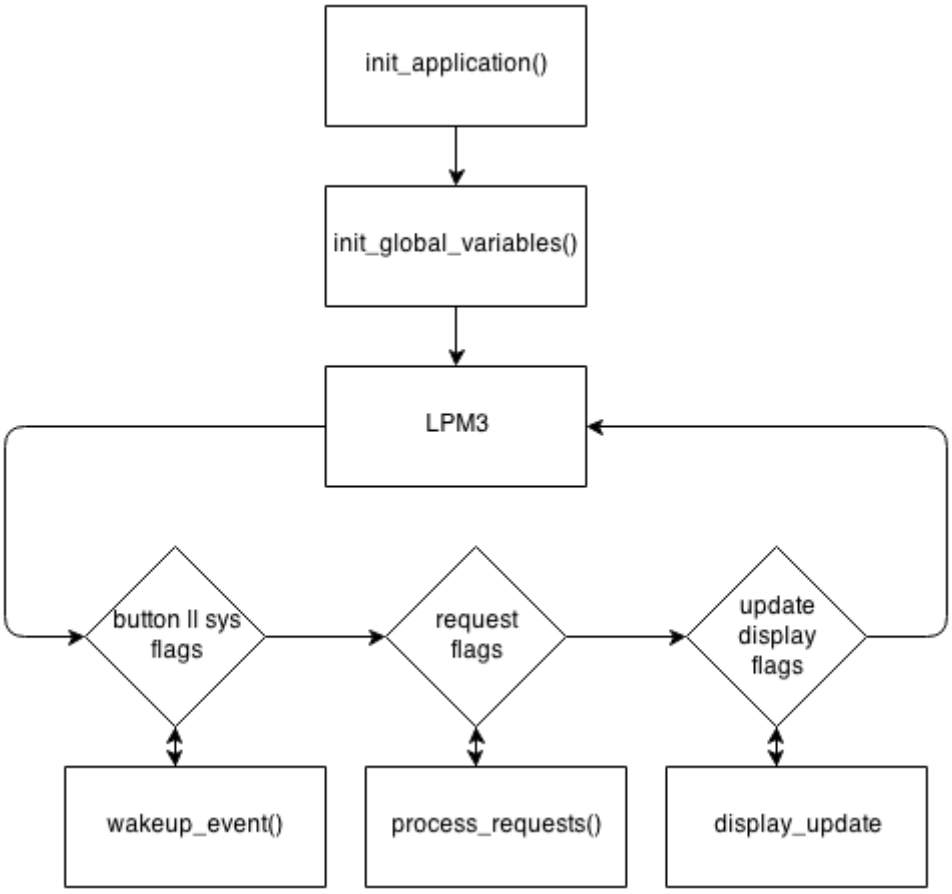

Figura 7 - Diagrama de flujo de main

Como se puede ver, el flujo de ejecución se encuentra durante gran parte del tiempo en el modo LPM3. Las siglas LPM3 se corresponden con *Low Power Mode 3*, es decir, se trata de un modo de bajo consumo en el que se apagan tanto la CPU como algunos de los periféricos. De esta manera se puede reducir la corriente de drenado del dispositivo y alargar la vida útil de la batería.

En init_application() se configuran las interrupciones de los temporizadores así como todos los módulos hardware del sistema, encendiendo posteriormente los que vayan a ser necesarios y dejando todos los demás apagados para ahorrar batería.

Con init_global_variables() se inician todas las variables globales, se leen y utilizan valores de calibración para los sensores, y se hace un reset de los módulos hardware desde el punto de vista lógico: se inician los punteros y variables propios de cada uno.

Tras la inicialización lógica del sistema se entra en un estado de bajo consumo para ahorrar batería, del cual sólo se sale cuando se produce alguna interrupción. Las fuentes de interrupción se listarán más adelante dentro de este mismo apartado. Una vez que se produce tal interrupción, dentro de su rutina de atención, se activan los *flags* propios de

cada rutina y se indica que a la salida de la misma se desactiva el modo de bajo consumo para poder procesar los *flags*, en caso de ser necesario.

Para el procesado de *flags* tenemos tres funciones. La primera es wakeup_event(), en la que se procesan los relacionados con los botones (decisiones a tomar en caso de la pulsación de alguno de ellos) y los relacionados con el sistema. La segunda se ocupa de peticiones, que son de tres tipos: medida de altitud, medida de la tensión de la batería y registro de datos en la memoria flash. La tercera y última tiene como propósito atender a los *flags* relacionados con el display, tanto de actualizaciones de una o ambas líneas como de mostrar mensajes de estado en la línea inferior.

Fuentes de interrupción

En el sistema hay definidas dos fuentes posibles de interrupción. Una se corresponde con la interrupción del puerto 2, al que están conectados los botones, y la otra es una interrupción periódica del temporizador TIMER0A.

La interrupción del puerto 2 tiene su rutina de atención a la interrupción en el fichero ports.c. En ella se comprueba primero si el reloj se encuentra en el modo de sincronización con el PC, en cuyo caso las pulsaciones se interpretarán como comandos a enviar al PC para controlar el ratón. En caso contrario la pulsación de un botón activará el *flag* correspondiente para ser procesado posteriormente en la función correspondiente en main.

La interrupción del temporizador TIMER0A se produce cada segundo. Cuando se activa se actualiza la hora, se solicita una actualización del display y, en caso de estar algún módulo encendido, se realiza una solicitud de medida o de registro de datos, dependiendo del modo en el que nos encontremos. De igual manera contabiliza el tiempo de forma que se puedan detectar pulsaciones largas de botones, tener un *timeout* para la luz, o para cualquier otro modo que lo precise.

Estructura del menú

Anteriormente se ha comentado cómo se navega por los distintos modos de ejecución. Sin embargo, el objetivo ahora es comentar cuál es la estructura del menú, esto es, qué opciones hay disponibles en cada elemento del menú. El menú se divide en dos líneas, una por cada línea del display. En cada una de las líneas tenemos los siguientes elementos disponibles:

- función directa, que no es la función que realiza por defecto cada elemento del menú, sino la complementaria. Por ejemplo, en el elemento del menú que indica la hora, es la función que cambia entre la hora y los segundos.
- función de submenú, activable mediante una pulsación larga del botón correspondiente. La mayoría de los modos no poseen una función de este tipo.
- función de display, que indica cuáles son los datos a mostrar para cada elemento del menú, así como para las funciones o subfunciones que se activen.
- función que trabaja en background; sea cual sea el modo en el que se esté ejecutando, hay que seguir actualizando la hora. Por este motivo todos los elementos del menú contienen esta función.

Almacenamiento de datos en la memoria flash

Como más tarde va a ser necesario conocer el formato en el que se almacenan los datos en la memoria, conviene comentarlo. Los datos se escriben en memoria en el fichero datalog.c, por lo que el primer sitio que consultar es ése.

Siguiendo el flujo de ejecución que seguiría el programa al iniciar el modo *DataLog*, los primeros datos que se escriben son una etiqueta de inicio de sesión, con valor 0xFBFF, y datos del registro: fecha y hora a la que se inició, modo de registro (qué datos se van a guardar) y con qué intervalo se va a hacer.
Posteriormente, en otra parte del código, y periódicamente (dependiendo del intervalo de registro) se guardan datos de los modos especificados en el modo de registro; para el ritmo cardíaco se guardan siempre 8 bits (un *byte*), y para la temperatura y altura como mínimo se guardan 12 bits (un *byte* y medio). El mínimo se especifica porque si nos encontramos en un modo en el que se registran temperatura y altura los datos de ambas se comprimen en tres *bytes* para permitir realizar un mayor número de registros. En caso de que no se registren a la vez temperatura y altura, cada uno ocupará 2 *bytes*. Al terminar la sesión de registro, se añade la etiqueta de fin de sesión, 0xFEFF. Una sesión de registro de pulsaciones, temperatura y altura, con cinco registros, tendrá el siguiente aspecto:

FBFF 0705 0A07 DE07 4601 F3E8 4B02 03E8 4901 E3E8 5501 93F3 5501 93F3 FEFF

Arquitectura del cliente para PC en el caso de referencia

El objetivo del cliente para PC es poder controlar algunos parámetros del reloj, así como permitir la descarga de datos. El software proporcionado por Texas Instruments, el *Chronos Data Logger*, tiene la siguiente interfaz de usuario:

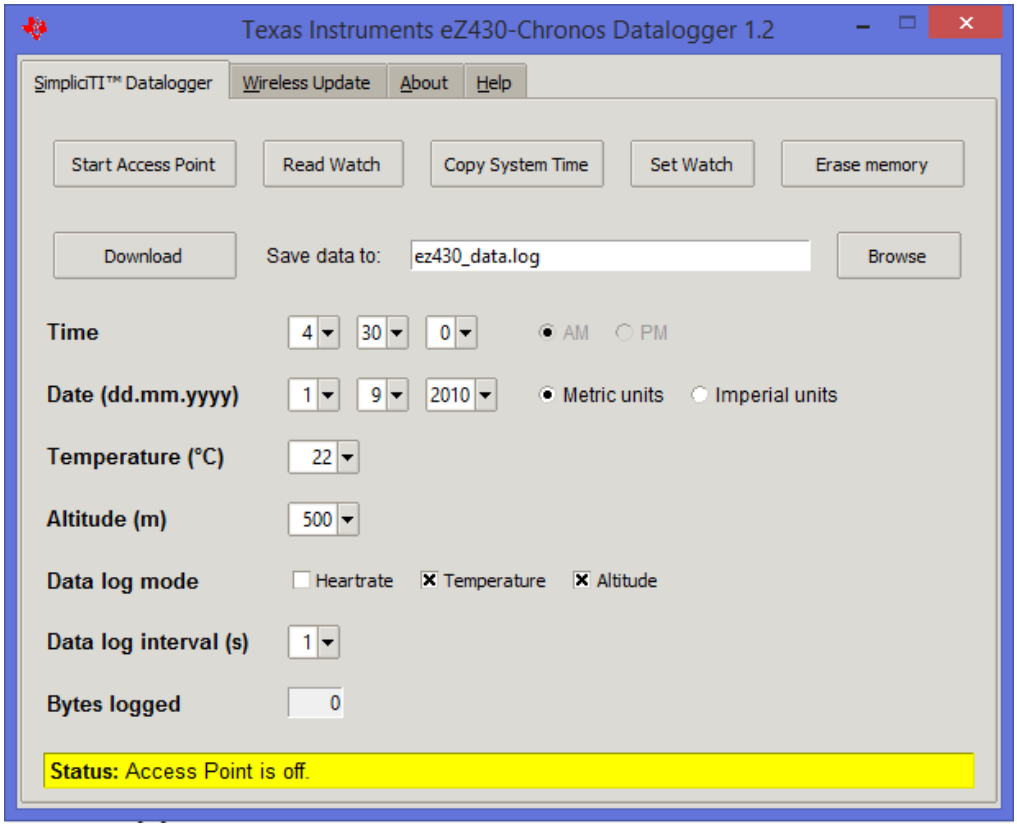

Figura 8 - Interfaz de usuario del Chronos Datalogger

Como se puede comprobar, el programa está orientado a eventos, reaccionando a los clics que se producen en los botones, entre eventos el sistema está en modo de bajo consumo para ahorrar batería. Cada botón activa una de las subrutinas disponibles.

De forma que la arquitectura de la aplicación de PC es la siguiente:

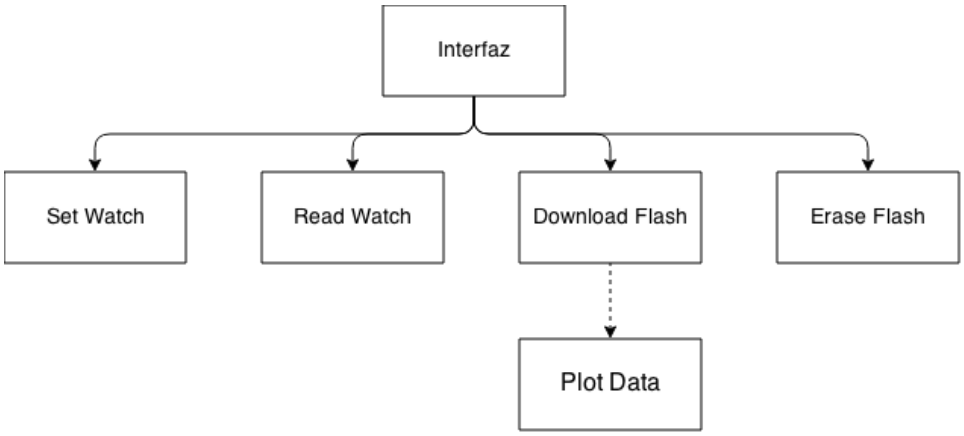

Figura 9 - Arquitectura de la aplicación de PC

Las líneas continuas indican cuáles son las funciones que están implementadas en la aplicación original, y la línea discontinua muestra una función que sería interesante añadir, ya que permitiría una lectura de las medidas más sencilla.

Puesto que entre el reloj y el PC está por medio el punto de acceso, es necesario, a pesar de haber analizado ya el código del microcontrolador, conocer cuál es el protocolo de comunicación del PC con el AP de cara a replicarlo. Dicho protocolo se analizará más adelante en el documento.

Desarrollo

Debido a una de las decisiones de diseño mencionadas anteriormente, el prototipo deberá ser capaz de gestionar automáticamente el proceso de adquisición y almacenamiento de datos, y posteriormente la conexión con el PC para comunicarlos al servidor. Esto conlleva dos implicaciones: es necesario un segmento más grande de código para el control de la ejecución, y se libera el uso de dos de los botones. Esta última implicación nos facilita la inclusión de otra de las funcionalidades deseadas, el botón de alarma, ya que ahora disponemos de botones para su implementación.

Aplicación cliente para PC

Para la programación del cliente para PC se ha utilizado, como se comenta más arriba, el lenguaje de scripting Python. Para la interfaz gráfica se ha hecho uso de la librería TkInter, una librería ampliamente utilizada. La interfaz de la aplicación con el PC queda de la siguiente manera:

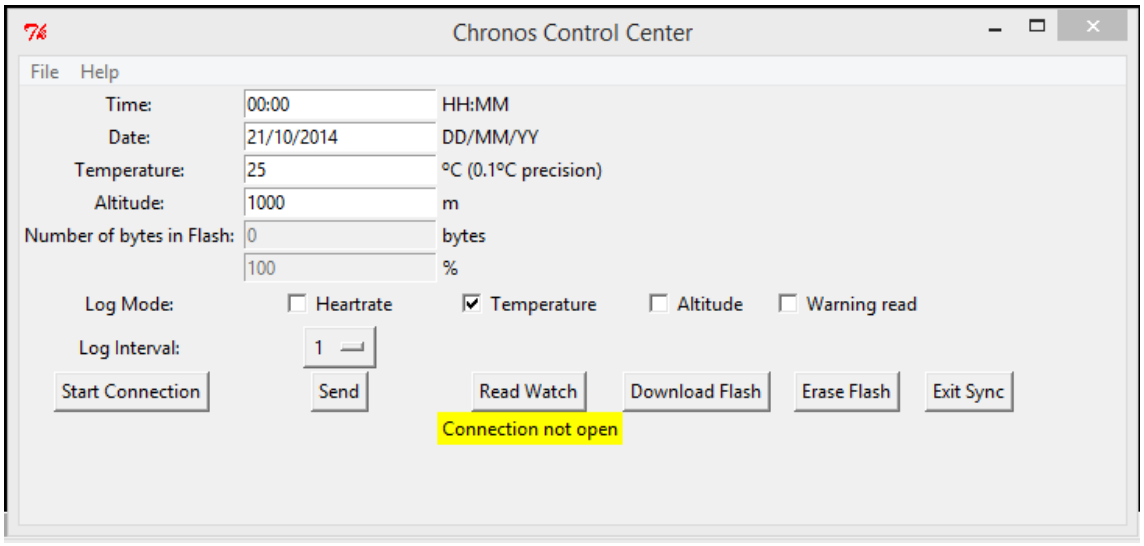

Figura 10 - Cliente para PC

Cada uno de los botones (menos el primero que únicamente sirve para abrir la conexión con el puerto COM del AP) realiza una función de comunicación con el reloj, que se explican más adelante. Para poder implementar dichas funciones es necesario primero comprender cuál es el protocolo de comunicación entre el PC y el AP, para posteriormente integrar las rutinas en el programa.

Protocolo de comunicación entre el PC y el Access Point (AP)

Para el análisis del protocolo se utiliza el programa USBPcap, como se ha comentado anteriormente. Con él se puede monitorizar el tráfico que circula por el puerto USB al que está conectado el AP. Una vez iniciada la captura de datos se pulsan uno a uno los botones del programa, de forma que se envían los comandos y los datos complementarios a esos comandos. De esta manera, se consiguen varios archivos de registro, uno por cada pulsación de cada botón, quedando los datos aislados de forma que sean distinguibles unos de otros, y del tráfico de fondo que mantiene la conexión con el AP activa.

El formato de trama que utiliza el protocolo es el siguiente:

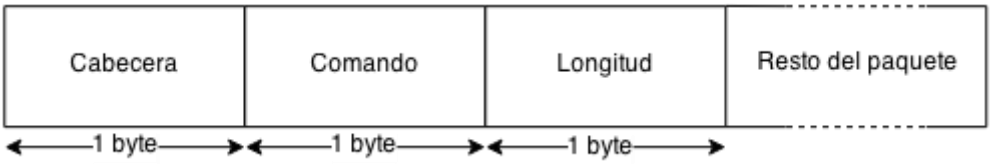

Figura 11 - Formato de trama del protocolo de comunicación

La cabecera siempre es un 0xFF, que marca el inicio de cada trama. El segundo *byte* incluye el comando con el que se le indica una acción al AP. El tercer *byte* marca la longitud total del paquete, incluyendo la cabecera. En el resto del paquete va información complementaria al comando, o información a transmitir hacia el reloj. Los comandos identificados son los siguientes:

- *Start Access Point*. Con él se inicia la comunicación con el punto de acceso.
- *Stop Access Point*. Con este comando se puede detener la comunicación con el AP.
- *Get Status*. Utilizado para conocer el estado de la conexión; conforma el tráfico de fondo.
- *Request Buffer Status*. Sirve para comprobar si se han recibido nuevos datos desde el reloj.
- *Read Buffer*. Una vez que se han recibido datos, se utiliza este comando para leerlos.
- *Send Command*. Este último comando es el que se utiliza cuando se quiere transmitir un comando al reloj.

Estos son los comandos que hay disponibles para controlar el AP. Los comandos que se pueden enviar al reloj son (estos comandos van contenido dentro del paquete, a partir del campo de longitud):

- *Set Watch*. Para seleccionar un valor para algunos elementos del reloj: hora, fecha, calibración de altura y temperatura...
- *Request Watch Data*. Para leer los datos comentados en el punto anterior.
- *Request Flash Data*. Se utiliza para leer los datos que estén almacenados en la memoria flash.
- *Erase Flash Data*. Con este comando se puede solicitar el borrado de la memoria flash.
- *Exit Sync*. Es el comando que se utiliza para abandonar el modo de sincronización.

Arquitectura de las rutinas

Aparte de conocer los comandos existentes hay que saber en qué secuencia se ejecutan, es decir, tanto el orden como el número de veces que se ejecuta cada uno.

Tras obtener las figuras representadas a continuación a partir de cada uno de los archivos de registro del tráfico por el puerto serie, se implementan en lenguaje Python:

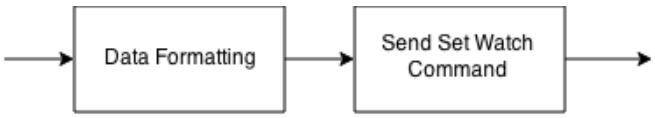

Figura 12 - Arquitectura de la rutina Set Watch

La arquitectura de la rutina *Set Watch* es bastante sencilla, se le da formato si es necesario a los datos contenidos en las cajas que se utilizan para introducir datos (ya que los datos que se envían al puerto serie tienen que ir en formato hexadecimal), y se manda el comando de actualizar los datos del reloj con los datos contenidos en las cajas como parámetros.

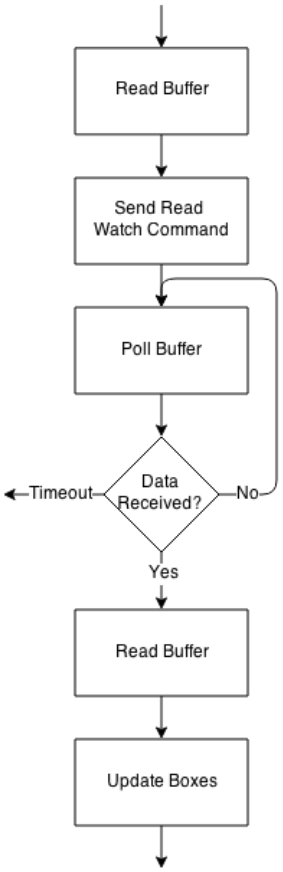

Figura 13 - Arquitectura de la rutina Read Watch

Para la rutina de leer datos el diagrama de flujo se complica un poco. Primero se lee el buffer ignorando lo que hay para vaciarlo. Tras mandar el comando para solicitar los datos del reloj se empieza a hacer *polling* al buffer de recepción. Mientras no se reciban datos se sigue mandando la consulta al buffer, hasta que haya pasado un cierto tiempo, tras el cual se sale de la rutina, o hasta que lleguen datos al buffer. Cuando llegan datos, se leen y se actualizan las cajas de la aplicación con los datos recibidos.

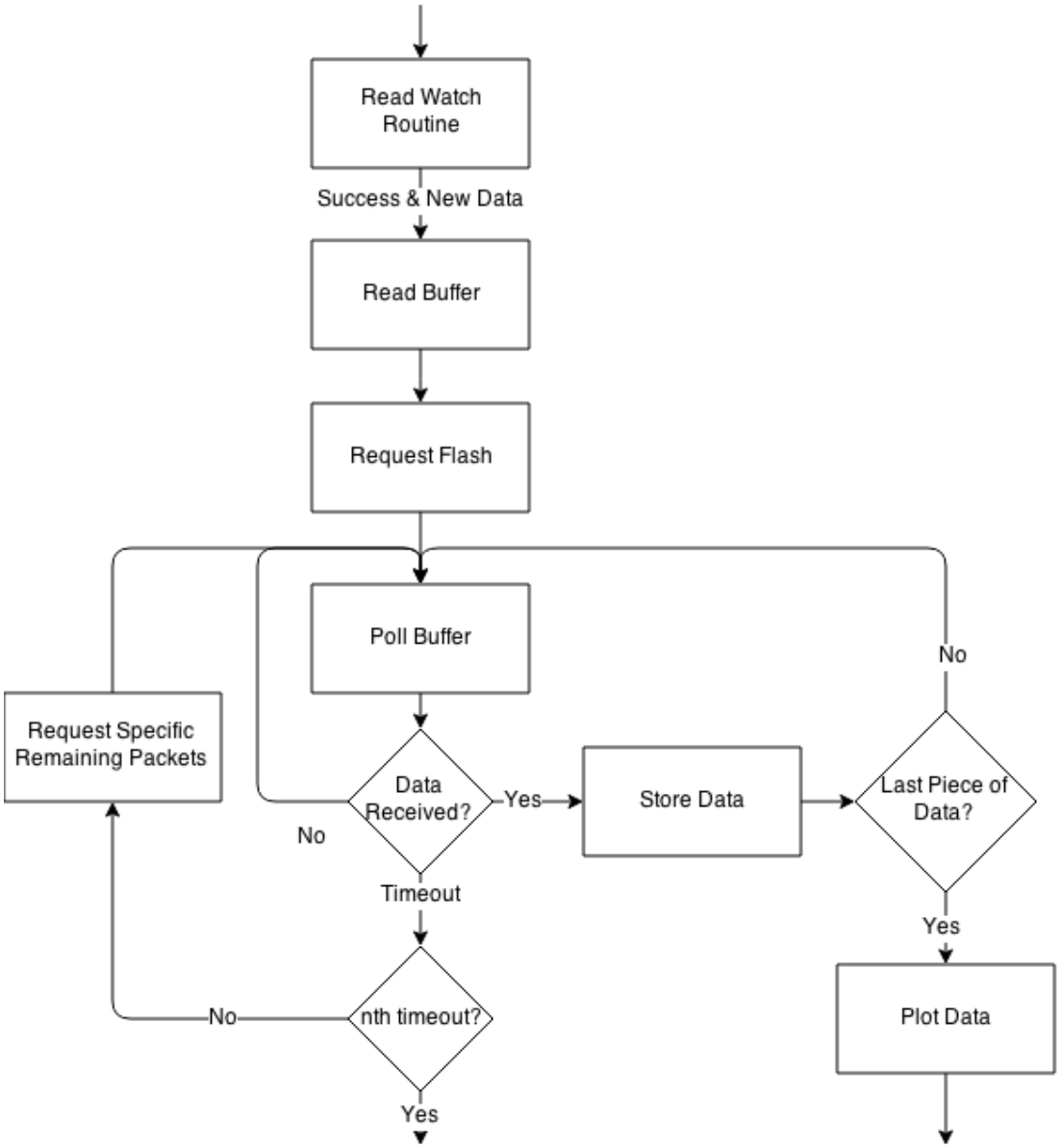

Figura 14 - Arquitectura de la rutina Download Flash

Para la descarga de datos hay que pasar primero por la rutina de lectura de parámetros del reloj. Uno de los datos que se envía es la cantidad de datos almacenados en la memoria Flash. Si la rutina *Read Watch* no se ejecute con éxito o no haya datos nuevos, no se continúa con la rutina. En caso contrario, se vacía el buffer leyéndolo y se manda el comando para la descarga de la memoria Flash.

Antes de continuar explicando el flujo de ejecución de la descarga conviene mencionar cómo se transmiten los datos de la memoria. Como se comentó en una sección anterior, los datos de la memoria se graban seguidos, únicamente separados por etiquetas cuando se cambia de sesión de registro. La memoria por tanto hay que transmitirla en paquetes más pequeños, ya que la longitud máxima del paquete está limitada. Por ello, se transmite en páginas de 16 *bytes*, que se agregan tras una etiqueta que marca el número de paquete que se envía. Con esta etiqueta se puede saber cuántos y qué paquetes faltan por ser recibidos, y cuando están descargados todos, ordenarlos.

Tras haber pedido los datos de la memoria Flash se inicia un *polling* al buffer para saber si se han recibido datos nuevos. En este punto pueden ocurrir dos cosas:

- Se recibe un dato, se guarda y se comprueba si es el último, ya que a partir de la información del estado del reloj recibida durante la rutina *Read Watch* sabemos cuántos paquetes tenemos que esperar. Si es el último, se escriben todos los datos en el archivo de texto y, como añadido al cliente de PC original, se representan en una gráfica integrada en la aplicación. Si no es el último, se vuelve a hacer *polling* al buffer a la espera del resto de paquetes
- No se recibe un dato durante cierto tiempo. Tras dicho tiempo, salta un *timeout*. Si es el primer *timeout* o el i-ésimo *timeout* dentro de los n permitidos, se envía al reloj un comando solicitando los paquetes restantes, volviendo a hacer *polling* al buffer. Este es también el recurso del que se hace uso cuando algún paquete se pierde y se tienen todos los demás, de forma que únicamente obtenemos los que necesitamos. Si es el n-ésimo *timeout* se sale de la rutina con un mensaje de error.

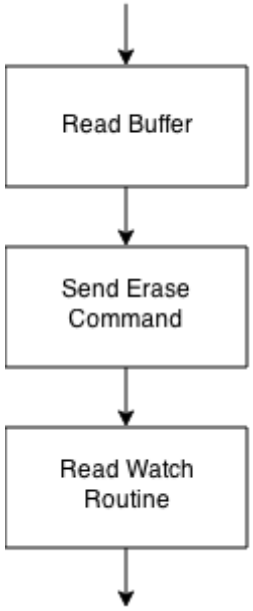

Figura 15 - Arquitectura de la rutina Erase Flash

Por último tenemos la rutina *Erase Flash*, que vacía el buffer, manda el comando de borrar la memoria Flash, y entra en la rutina *Read Watch*. De esta forma, cuando las cajas se actualizan, el usuario puede comprobar cuánta memoria tiene almacenada el reloj, e interpretar que ha sido satisfactoria la recepción del comando si la cantidad de memoria almacenada es 0 *bytes.*

Algoritmo para Detección de caídas

Para detectar caídas se dispone de un acelerómetro comercial con varios modos de funcionamiento[6]:

- Medida. En este modo el acelerómetro proporciona datos de la aceleración a través de la interfaz SPI/I2C. Se genera una interrupción por el pin INT cada vez que hay una nueva medida disponible para su lectura.
- Detección de movimiento. Este modo está pensado para ahorrar batería en el sistema. En este modo, cuando se detecta movimiento se genera una interrupción por el pin INT. Una vez que se ha producido la interrupción se puede leer en un registro cuál ha sido el eje que ha generado la interrupción. Además, se puede configurar para cambiar al modo de medida según se genere la interrupción para iniciar las medidas.
- Caída libre. Cuando se configura el acelerómetro en en este modo, se genera una interrupción cuando se detecta la caída libre; no es posible cambiar automáticamente al modo de medida si se desea, así que habría que reconfigurarlo mediante software.

El modo de medida no parece el más adecuado para detectar una caída, ya que genera una interrupción periódicamente para la lectura de datos, lo que nos supone dos inconvenientes; el primero es que habría que hacer un procesado de datos en el sistema, y sería complicado ya que la memoria disponible no es muy grande. Y el segundo es en el tema de batería, porque cada vez que hubiera datos nuevos se saldría del modo de bajo consumo para atender la interrupción.

A primera vista, el modo de caída libre es el que más atrae para la función deseada. Echando un vistazo a la hoja de características, se ve que es inviable. En la siguiente gráfica se comprueba por qué:

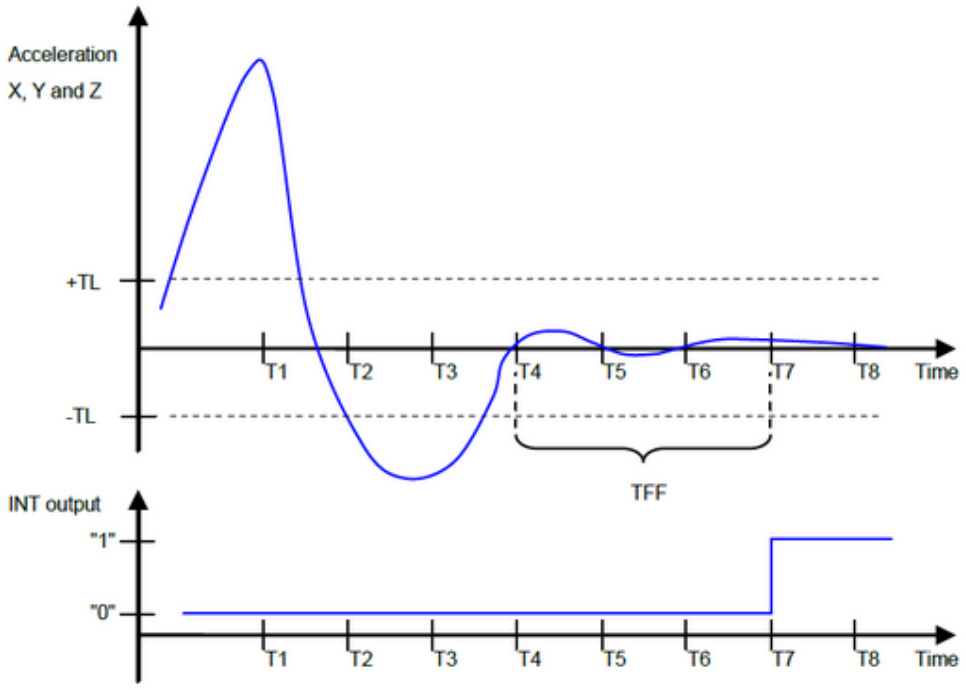

Figura 16 - Comportamiento del acelerómetro en caída libre

La interrupción se genera cuando pasado un tiempo (configurado por el programador) la aceleración se encuentra dentro de un umbral (también definido por el programador) alrededor de cero, tras haber sufrido una aceleración y haber sobrepasado el umbral, lo cual quiere decir que primero debe acelerar para iniciar la caída y luego anularse la aceleración. Para que en caída se anule dicha aceleración debe estar, efectivamente, en caída libre. Esto significa alcanzar la velocidad terminal, para lo cual hace falta un tiempo de caída demasiado grande, y no tiene sentido para ser utilizado en la detección de caídas. Por otro lado, se puede pensar en intentar ajustar los valores del umbral y el tiempo de detección, de forma que se genere la interrupción por haber estado por encima del umbral en un tiempo pequeño. Sin embargo, un umbral demasiado bajo significa generar muchas alarmas falsas tras cualquier movimiento, y un umbral demasiado alto significa no generar una interrupción siquiera con la caída. Esto se debe a que una persona cuando se cae el acto reflejo que tiene es intentar agarrarse a cualquier superficie para evitar la caída, de modo que se genera aceleración y no se llega a mantener entre los dos umbrales para generar la interrupción.

Queda por tanto, como última opción, el modo de detección de movimiento. Se comprueba su funcionamiento en la siguiente figura:

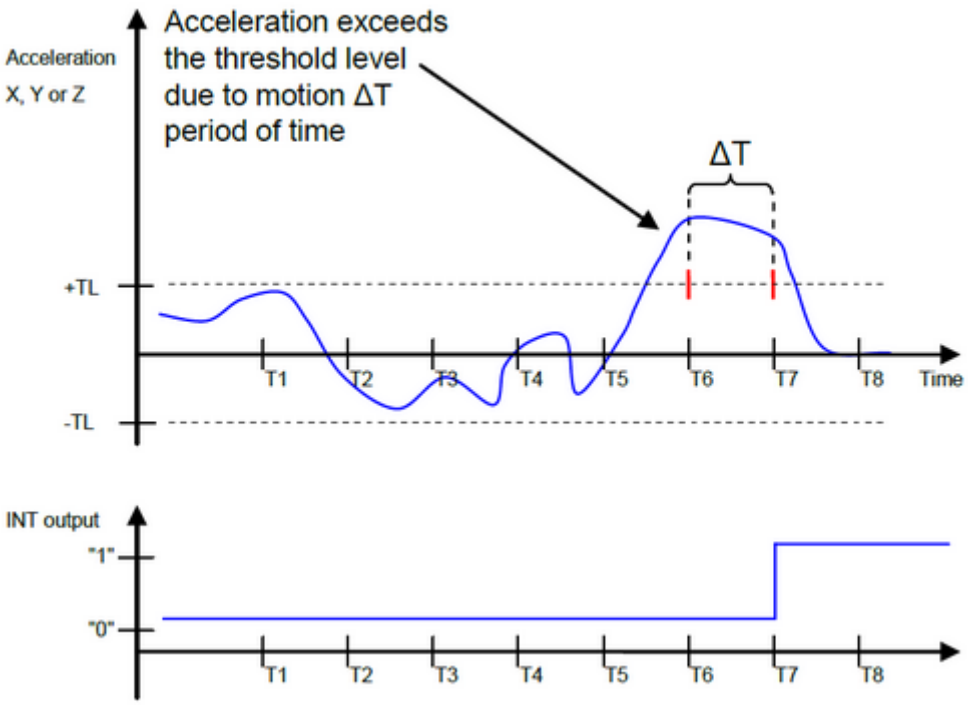

Figura 17 - Comportamiento del acelerómetro en detección de movimiento

El programador puede definir dos umbrales, uno indicando el nivel que debe de sobrepasar la aceleración en cualquiera de los ejes para generar la interrupción por el pin INT, y otro que define el tiempo que debe estar por encima de dicho umbral para que se genere. Configurando un umbral alto los movimientos que se detectan son los bruscos, tales como los producidos por una caída. Teniendo en cuenta además que el producto final sería utilizado por personas mayores (con movilidad reducida), no se generarán muchas falsas alarmas. Respecto al registro que contiene el tiempo que debe estar sobre el umbral, el tiempo mínimo que se puede configurar es 100 ms y el máximo 700 ms, valores suficientes para la aplicación deseada.

De forma que, siendo el modo de detección de movimiento el modo seleccionado, el siguiente paso es configurar el acelerómetro para este modo. Los registros, todos ellos de 8 bits, a los que debemos prestar atención son:

- RSTR (dirección 0x04); resetea el acelerómetro si escribimos 0x02, 0x0A, 0x04 en este orden.
- CTRL (dirección 0x2); registro de control. En él se configuran aspectos como el modo de funcionamiento, el nivel de interrupción, etc. Se configura como sigue (código 0x88):
    - G_RANGE; se selecciona el rango de 2g para tener una mayor precisión.
    - INT_LEVEL; por cómo estaba configurada la detección de interrupción en el microprocesador, se configura aquí la generación a nivel alto.
    - MDET_EXIT; tras haber detectado movimiento, el acelerómetro cambia al modo de detección de movimiento.
    - I2C_DIS; su valor es indiferente ya que para la comunicación con el acelerómetro se usa la interfaz SPI.
    - MODE; 100 es el valor que debe tener para estar en el modo de detección de movimiento
    - INT_DIS; se desea tener el bit a nivel bajo para activar interrupciones.
- INT_STATUS; cuando se ha generado una interrupción sin estar en el modo de medida de aceleración, se puede consultar en este registro cuál ha sido la fuente de interrupción. Los dos bits inferiores nos indican en cuál de los ejes se ha sobrepasado el umbral y por tanto generado la interrupción. Leer este registro lo resetea. Se encuentra en la dirección 0x05.
- DOUTx; en las direcciones 0x06, 0x07 y 0x08 se pueden consultar los valores medidos en los ejes X, Y y Z, en complemento a dos.

- MDTHR; en este registro se almacena el umbral para la detección de movimiento. Se encuentra en la dirección 0x09, y el valor que se ha elegido para este umbral es 0x38. Cada bit tiene un peso, indicado en la siguiente tabla, extraída del *datasheet*:

Range	G_RANGE	B7	B6	B5	B4	B3	B2	B1	B0
2g	1	x	x	571	286	143	71	x	x
8g	0	x	4571	2286	1142	571	286	143	1/14 = 71 mg

x=not used bit

Figura 18 - Tabla MDTHR

- MDFFTMR; registro en el que que se configura el tiempo que debe estar por encima del umbral. Está en la dirección 0x0A, y nos interesan los cuatro bits inferiores.

Bits	Mode	Initial Value	Name	Description
7:4	RW	3h	MDTMR[3:0]	Motion detection timer bits
3:0	RW	3h	FFTMR [3:0]	Free fall detection timer bits

Figura 19 - MDFFTMR

- El valor por defecto es 0x03, que equivale a 300 ms, ya que el LSB indica cuántas fracciones de 1/frecuencia[Hz] se esperan.

Timer	MDTMR			
Register bit number	B7	B6	B5	B4
Timer bit number	MDTMR b3	MDTMR b2	MDTMR B1	MDTMR b0
CMA3000-D01, MODE bits 100 ODR: 10Hz	x	400	200	1/10s = 100 ms

Figura 20 - Frecuencia determinada por el MDTMR

Para el tratamiento de la interrupción en el programa se han definido tres *flags* a nivel global dentro del fichero project.h, que todos los demás archivos incluyen, en el *struct* que contiene los *flags* de sistema. Dichos *flags* son:

- *as_motion*; este *flag* es el que se activará en la rutina de interrupción para indicar que ha habido una detección de un movimiento brusco, para su posterior procesado.
- *warning*; una vez que se procesa el *flag as_motion* se activa este otro *flag*, que es el que indica que es posible que sea necesario avisar a alguien. Este *flag* es el que se usará también después para el botón de alarma.
- *warning_timeout*; servirá de contador para saber cuánto tiempo lleva la alarma activada, y en caso de superar cierto umbral de guarda (para que el usuario pueda desactivarla si se trata de una falsa alarma), se pasaría a la comunicación con el PC para avisar del problema.

Configurada la interrupción en el lado del acelerómetro, queda hacer lo mismo en el lado del procesador. Consultando las conexiones, el pin INT está conectado a un pin del puerto 2[9], al igual que los botones. Por tanto, la atención a la interrupción generada cuando se detecte una caída se tratará en la misma rutina que la que se genera cuando se pulsa un botón. Dicha rutina se encuentra en ports.c, como se ha comentado en este post en el apartado de fuentes de interrupción.

De modo que dentro de la rutina PORT2_ISR se incluyen unas líneas de código para leer el pin asociado al pin INT del acelerómetro, para hacer una lectura del registro 0x05 (INT_STATUS) que limpie el *flag* de la interrupción, y para activar el *flag as_motion*.
Dentro del código de main, en la rutina que se encarga del tratamiento de los eventos (wakeup_event) se añade código para el procesado de los *flags* de alarma: se desactiva *as_motion* y se activa *warning*. De esta manera se desacopla la interrupción por caída del contador de tiempo de alarma, que es común a las caídas y a la alarma de usuario.
Para controlar cuánto tiempo se encuentra activo el *flag warning* es necesario utilizar la única interrupción periódica del sistema: la producida en *timer* cada segundo. Dentro de la rutina de atención a la interrupción de *timer* se comprueba si está activo el *flag*, en caso afirmativo se incrementa el contador de tiempo *warning_timeout* y se activa el *buzzer* para que el usuario y personas en la cercanía se enteren de que se ha encendido la alarma. Además, si el contador supera un cierto valor umbral WARNING_TIMEOUT se entraría al modo de sincronización con el PC para avisar de la alarma (con valor 10 segundos, un valor suficiente como para dar tiempo a desactivarla si fuese falsa alarma, pero que aun así no es demasiado alto).

Botón de alarma

La forma de implementar un posible botón de alarma ha sido utilizando la pulsación simultánea de dos de los existentes (botones NUM y DOWN). Con el objeto de detectar ambas pulsaciones se ha modificado el código, ya que aunque está previsto detectarlas (para el bloqueo de los botones), en la práctica no está implementado.

Dentro de la rutina de atención a la interrupción del puerto 2 se contemplan las pulsaciones individuales de botones, pero no las pulsaciones de dos a la vez. Esto es, está implementado con estructuras if - else if, de forma que si entra a uno de ellos no entra a los demás, y por lo tanto no se activa el *flag* correspondiente al segundo botón pulsado. Añadiendo la posibilidad de marcar un *flag* de uno de los botones dentro del if del otro botón se corrige esto.

Para evitar pulsaciones involuntarias e intentar reducir la cantidad de falsas alarmas, la alarma se tendrá en cuenta cuando se realice una pulsación larga de los botones, es decir, cuando se mantengan pulsados los dos botones durante un tiempo definido (en este caso, 2 segundos). Al igual que en la detección de caídas, se utiliza el temporizador para contabilizar el tiempo que llevan presionados. Sin embargo, en esta ocasión, se utiliza una función que ya tenemos definida en la demo.

Una vez que ha pasado el tiempo correspondiente se comprueba si ya estaba en modo de alarma. En caso de estarlo, la corta, porque significaría que ha habido una falsa alarma y el usuario no necesita atención. En el caso de que no lo esté, se activa el *flag warning*.

Flujo de ejecución automático

Para automatizar la ejecución hay que analizar primero cuál es el funcionamiento deseado. Lo ideal sería que una vez arrancado el programa (ya sea por ser la primera conexión o por cambio de pila) permitiera al técnico ajustar la hora actual y, si fuera necesario, algún otro parámetro. Posteriormente, dejar el programa del cliente de PC corriendo y que el reloj de manera automática cambiara entre el modo *DataLog*, cuando la memoria estuviera llena, y el modo *Sync* cuando la memoria estuviera vacía.

Así pues, los cambios a realizar son aquellos que permiten que el reloj esté a la espera hasta que el operario da el visto bueno a los parámetros, momento en el que permite al reloj empezar la ejecución automática. Hay que encontrar las condiciones que definen los cambios de estado; dos de ellas han quedado claras al exponer el modo de funcionamiento: la primera, cuando la memoria esté llena, y la segunda, cuando la memoria esté vacía.

Para realizar los cambios, lo más conveniente, manteniendo el modo de funcionamiento original hasta cierto punto, es tener unos *flags* globales definidos en project.h que se cambien en alguna parte del programa y se procesen en main, ya que los cambios de los *flags* van a producir un wakeup del sistema. Por tanto, se definen dos nuevos *flags* que servirán para marcar los cambios.

El punto en el que marcar el cambio del modo de registro de datos al modo de sincronización es claro, ya que se produce cuando la memoria está llena, y eso ocurre

únicamente en *DataLog*, en la parte de registro. Es tan sencillo como buscar el punto en el que se marca la memoria como llena; el *flag* se debe activar tras haber escrito la etiqueta de fin de sesión.

Para el cambio entre el modo *Sync* y el modo *DataLog* hay que analizar los archivos que utiliza cuando se encuentra en el modo sincronización, ya que utiliza una librería que viene compilada y ensamblada y se comunica con el programa mediante *callbacks*. Hay dos funciones de *callback*; una es para decodificar comandos del punto de acceso (con una sentencia switch) y otra para decodificar datos provenientes del PA. Considerando los posibles comandos, se comprueba que hay uno de salida del modo *Sync*. Por tanto, cada vez que nos encontremos en el modo *Sync* y queramos salir de él podemos enviar el comando correspondiente desde el cliente de PC, lo cual tiene sentido ya que en esa comunicación el reloj es *slave* y el PC es master, de forma que es el maestro el que decide cuándo terminar la comunicación. Controlando de esta manera la salida del modo de sincronización se resuelve además la salida del modo de sincronización inicial, la que tiene lugar tras el cambio de la pila o tras el arranque inicial del reloj. El *flag* de cambio se modifica, por tanto, en la función de decodificar *callbacks*, en el comando de salida del modo *Sync*.
De modo que las modificaciones a aplicar son:

- Eliminar la respuesta a las pulsaciones de los botones *NUM* y *DOWN*, así como el código asociado al tratamiento de su pulsación, para quitar el control al usuario de las funciones del menú inferior. Las acciones que se deben desactivar son las pulsaciones individuales, no así las pulsaciones simultáneas, que son las que se deben tratar para la alarma de usuario.
- Arrancar el reloj en modo *Sync* para su primera configuración. La llamada a la función que realiza el inicio del modo se debe introducir justo tras la inicialización de todos los módulos hardware y las variables globales, y antes de la entrada al bucle infinito, de forma que a la entrada nos encontremos ya en el modo con su función principal activada.
- Activación de los *flags* en los puntos mencionados anteriormente y tratamiento de los mismos, con el objetivo de cambiar lo que se muestra en el menú, activar o desactivar los módulos radio según corresponda, y realizar el cambio entre los modos de registro y sincronización.

Además, para informar de la detección de una alarma (ya sea por caída o porque el usuario desee activarla) así como del nivel actual de la batería (medido), se incluyen dos líneas dentro de la decodificación del *callback*. Dentro de todos los datos que se envían al PC - que se pueden analizar en el código en C, pero se comentarán posteriormente en la sección del cliente de python - hay tres *bytes* que se envían siempre a cero, porque llevan información sobre funciones que no se usan en la demo de registro de datos. Aprovechando que esos *bytes* están sin utilizar se enviarán ahí el nivel del *flag* de alarma y el nivel de la batería medido en su función correspondiente.

Resultados

En esta sección se presentarán los resultados obtenidos tras el desarrollo del proyecto, tanto lo referente al software en el controlador como en el apartado del cliente para PC.

Software en el controlador

Tras haber realizado los cambios mencionados en la sección de desarrollo, se comprueba que son funcionales, pero a medida que se van integrando los módulos el sistema da problemas.

Para hacer uso del botón de alarma es necesario mantener los dos botones inferiores durante dos segundos, de forma que se evita que las pulsaciones involuntarias activen la alarma. Cuando se hace uso de esta funcionalidad, y tras haber pasado 10 segundos en los que el usuario puede desactivarla en el caso de haber sido activada por error, en el PC se recibe un bit que indica el evento y se genera un aviso. Con una nueva pulsación se desactiva la alarma, cumpliendo con los requisitos definidos para esta funcionalidad.

En el apartado de detección de caídas, se detectan satisfactoriamente. Los movimientos pausados (que se suponen propios de ancianos y personas con movilidad reducida) no provocan la activación de la alarma, mientras que movimientos más bruscos como podría ser el de una caída sí que la activan. Una vez que se detecta, el usuario tiene 10 segundos para desactivarla mediante el uso del botón de alarma. En el caso de que el usuario no lo haga, el reloj comunica correctamente la caída al PC, que genera un aviso de manera correcta.
En el caso de la ejecución automática, el inicio se realiza de la forma esperada, pero a partir de cierto punto en la ejecución (en concreto, en el punto en el que debe pasar del modo de registro de datos de vuelta al de sincronización) el programa no responde de la manera esperada y se queda en un estado del que no es posible salir al no tener control sobre los botones que lo permitirían.

Software en el PC

En el PC se implementan correctamente las rutinas. La descarga de datos ya no sea realiza en un archivo cuyo nombre lo elige la persona que opere el PC, sino que ahora se descargan todos a un mismo archivo, colocando los datos nuevos a continuación de los antiguos pero manteniendo cierta separación que permite diferenciar entre distintas sesiones de descarga. De esta forma, están todos los datos accesibles en un solo archivo y se evita la eliminación de archivos existentes.

También está implementada la funcionalidad de representar los datos de la memoria Flash tras una sesión de descarga, así como avisos tanto de alarma como de batería baja mediante una ventana emergente cuando se detectan a través de los bits enviados en la trama de estado.

La ejecución automática se implementa tras la pulsación de uno de los botones del menú, para así permitir una primera configuración del reloj manual y a partir de ahí la ejecución normal.

Pruebas y experimentos

Caídas

Para la prueba de detección de caídas se han realizado dos experimentos. Uno ha consistido en dejar caer el reloj desde distintas alturas para comprobar a partir de qué altura la alarma se activa. Dadas las diferentes alturas, el comportamiento del reloj al caer al suelo ha variado. Se ha realizado estudios teniendo este dato en cuenta, ya que en algunos casos el reloj caía directamente al suelo mientras que en otros giraba durante la caída por lo que se detectaba este movimiento en distintos ejes del acelerómetro, pudiendo no llegar alguno de ellos al umbral de alarma. Después de varios lanzamientos, los resultados muestran que la alarma se activa con la caída desde aproximadamente metro y medio, cuando el reloj no da vueltas en el aire. En algunos casos en los que el reloj describió vueltas en el aire en ninguno de los ejes se alcanzó la aceleración necesaria para constituir una detección exitosa.

El segundo experimento ha consistido en entregarle el reloj a una persona que se ha tirado al suelo simulando una caída, para comprobar que la caída se detecta correctamente. Dicho sujeto mide 1.75 metros de altura, y en él queda el reloj suspendido a una altura de aproximadamente 1 metro en posición de reposo.

Este sujeto ha simulado distintos tipos de caídas como por ejemplo, caerse al suelo en una zona despejada (donde no tenía objetos al alcance con los que frenar su caída.), caerse sobre mobiliario (donde sí que disponía de los lugares a los que asirse y por lo tanto cambiaba el perfil de la aceleración) y movimientos tanto rápidos como lentos por parte del sujeto que no se consideran caídas. En el primer caso, la alarma se activaba satisfactoriamente en gran parte de los casos, ya que la aceleración de la caída se veía incrementada por el acto reflejo del usuario al bajar los brazos para intentar frenar la caída. En los casos en los que falló, la suposición es que al caer de lado el movimiento que realiza la mano provoca que no se alcance el umbral con el que se activa la alarma.

En el segundo, la caída se activaba de manera efectiva en la mayor parte de los casos, siendo aquellos en los que no se activaba los correspondientes a ciertos giros de la muñeca al agarrarse que provocaban un nivel inferior al umbral en los ejes del acelerómetro. Para movimientos rápidos, la alarma se activaba siempre, para movimientos de una amplitud relativamente pequeña (el equivalente a estar de pie y tener el brazo en posición de reposo y subirlo hasta la cabeza). De igual manera, se ha comprobado que con movimientos lentos (los esperados en una persona mayor) no se activa. A continuación quedan reflejados los resultados obtenidos a partir de la realización de 20 pruebas para cada caso:

- 85% de éxito en la detección de caídas hacia delante en una zona despejada.
- 20% de éxito en la detección de caídas laterales en una zona despejada.
- 65% de éxito en la detección de caídas de espaldas con lugares a los que asirse.
- 100% de activación de alarma para movimientos rápidos del brazo.
- 15% de activación de alarma para movimientos lentos del brazo.

Los resultados obtenidos son los esperados, sabiendo de antemano que los movimientos del brazo podían alterar el resultado de las medidas de la aceleración, y por tanto, de la

detección de caídas. Respecto de las caídas, muchas son frontales, debido a tropiezos, y las caídas laterales no son tan frecuentes como las frontales, por lo que el sistema podría aplicarse para el control de las caídas de personas mayores en un entorno doméstico, en el que el sistema pudiera comunicarse inmediatamente con el PC. En entornos fuera del hogar perdería la utilidad ante la inhabilidad para comunicarse inalámbricamente con el cliente de PC.

Las caídas laterales tienen tan poco éxito debido a los movimientos del brazo durante la caída en un acto reflejo por intentar frenarla. En los casos de caída frontal, el brazo se mueve en la misma dirección de la caída (hacia adelante), mientras que en las caídas laterales se gira el tronco en un intento de poner ambas manos en el suelo antes de golpear con el resto del cuerpo. El hecho de que el brazo se mueva con un patrón distinto de aceleración provoca que la aceleración no sea la suficiente para alcanzar el umbral de alarma.

Una posible solución para homogeneizar los casos de éxito en la detección de caídas e independizarlos del tipo sería medir la aceleración en algún punto del cuerpo que no tuviera tanta movilidad y variación durante la caída como el brazo. Un posible punto favorable a la hora de colocar el acelerómetro sería la parte baja de la columna, ya que el recorrido es aproximadamente igual independientemente del sentido de la caída.

Facilidad de uso

De cara a comprobar la facilidad de uso, se le ha entregado el dispositivo con la versión del firmware que realiza la ejecución automática a un sujeto para que lo utilizara durante una sesión de registro, sin haberle informado previamente de su funcionamiento. El sujeto tardó aproximadamente 5 minutos en entender el funcionamiento del dispositivo. Comenzó pulsando los botones de manera arbitraria hasta que entendió cuál era el botón de menú (el botón superior izquierdo). A continuación, en cada opción del menú pulsó los otros tres botones, para descubrir que únicamente otro botón estaba activado, y las opciones que le ofrecía este botón restante. Con él, averiguó qué funciones podía realizar en ese submenú (las correspondientes a cambiar entre hora y segundo, y día/mes y año). El sujeto entendió rápidamente la estructura cíclica del menú, de manera que rápidamente se sintió suficientemente confiado como para dejar de investigar sobre su funcionamiento.

Registro de datos

Para el registro de datos se ha recurrido al sujeto de la prueba anterior pero se ha cambiado en este caso el firmware a la versión en la que el usuario tiene control sobre todas las funciones (pero que incluye también la alarma de usuario e interrupción por caídas), por problemas con la ejecución que se comentan más adelante. En este caso, y habiendo explicado ya el funcionamiento de las nuevas funciones sobre las que adquirió control, se realizaron varias sesiones de registros, comprobando con la aplicación proporcionada por Texas Instruments que el registro de datos se realizaba de manera satisfactoria y los datos obtenidos eran coherentes. De esta manera, se consiguió un set de datos válido y con el que poder comparar con los resultados que se obtendrían en pruebas posteriores.

Descarga y representación de datos

Tras la sesión de descarga, se realizaron varias descargas de ese set, para comprobar la cantidad de fallos producidos a la hora de descargar los datos y si los datos descargados y escritos en el archivo de texto se correspondían con los almacenados en el reloj (de los que había constancia con la prueba anterior), además de monitorizar la actividad en el puerto USB. Los resultados obtenidos muestran que en aproximadamente un 70% de las ocasiones la descarga fue satisfactoria. En estos casos, la descarga se realizaba transmitiendo todos los paquetes seguidos, y no se perdía ningún paquete. Una vez que se completa la descarga se procede a grabar los datos descargados en el disco, concatenándolos a los ya existentes en caso de haberlos. Posteriormente, se representan los datos de la descarga, consiguiendo una gráfica como la que se representa a continuación.

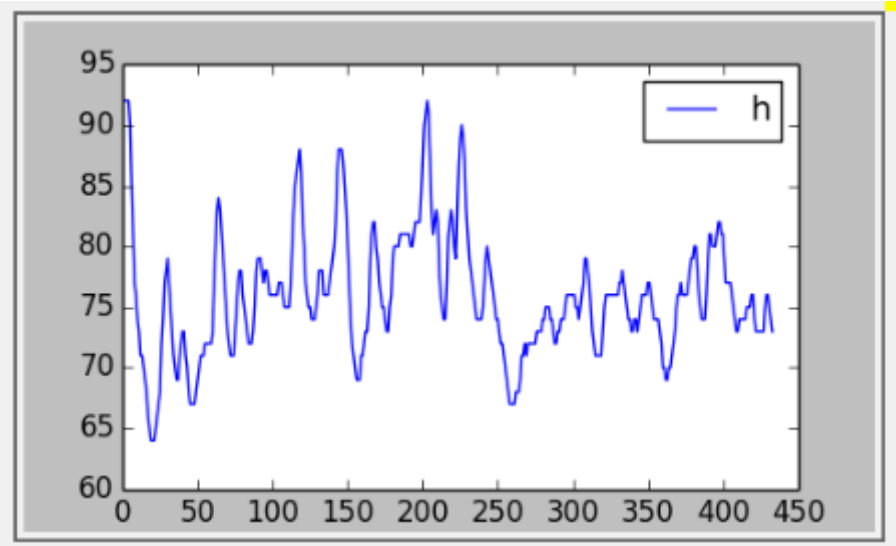

Figura 21 - Descarga de la memoria Flash

La interfaz gráfica completa tras la descarga queda como se muestra a continuación:

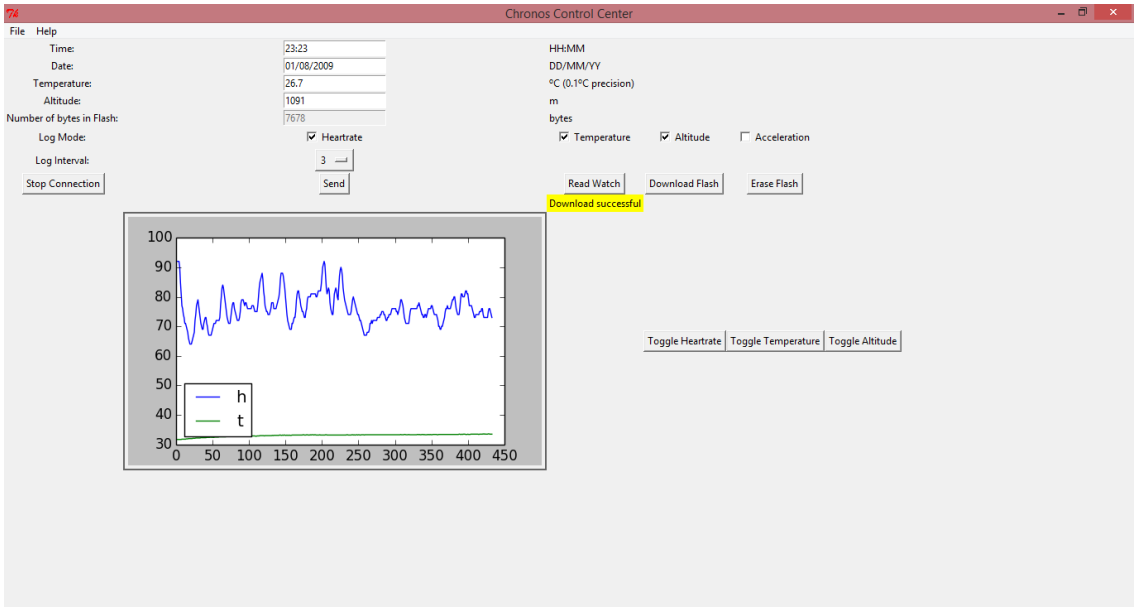

Figura 22 - Aplicación tras una sesión de descarga

Se han añadido tres botones para poder elegir qué datos se muestran y qué datos permanecen ocultos; los tres botones se corresponden con cada uno de los datos que es capaz de registrar el reloj, pero en caso de poder aumentar el número de sensores, ampliar la cantidad de botones supone un cambio muy sencillo.
En los casos en los que no se ha ejecutado correctamente, se perdía alguno de los paquetes y a la hora de pedir ese paquete en concreto el reloj no respondía de la manera esperada. El AP dispone de un LED que indica cuándo está ocupado comunicándose con el PC. Se ha observado que si se inicia la descarga cuando el LED está encendido (el AP ocupado) la descarga es errónea. En el resto de casos la descarga fue satisfactoria. Por ello, la suposición para la pérdida de paquetes es que el tráfico de fondo provoca un retardo en la comunicación que hace que esos paquetes no se reciban. Respecto de la imposibilidad de recuperar los paquetes perdidos, no se han producido suficientes casos con la aplicación que viene por defecto (pocos casos y para pocos paquetes en todos ellos) como para poder comparar y saber cuál es el origen del fallo. Además, la no existencia de documentación sobre el proceso de comunicación entre el AP y el PC dificulta su resolución.

Flujo de ejecución automático

Se han realizado pruebas de la ejecución tras haber implementado el flujo de ejecución automático. El resultado de la ejecución no es el esperado, no consiguiendo sincronizarse con el reloj tras terminar la sesión de descarga. Para intentar determinar el origen del fallo se realiza la ejecución paso a paso mediante el *debugger* del entorno de desarrollo. A través de esta ejecución paso a paso se puede comprobar que, al tener la optimización configurada al máximo, las llamadas a funciones se integran como parte del código tras el punto en el que se realiza la llamada, los bucles se desenrollan, etc., haciendo más lenta la llegada al punto en el que empieza el funcionamiento erróneo. No se ha podido determinar el origen del problema puesto que el entorno de desarrollo se bloquea tras algunos pasos y obliga a iniciar la ejecución desde el principio, por lo que no se alcanza el punto del bloqueo.

Conclusiones

En primer lugar, cabe mencionar que las WBAN como tecnología tienen un gran interés tanto para los pacientes como para los gobiernos, ya que permite llevar una vida más cómoda a los pacientes y ahorrar en sistemas de monitorización y procedimientos de diagnóstico que en la actualidad se pueden proporcionar únicamente en un hospital o centro de salud, aparte de reducir las esperas en las consultas médicas. Por otro lado, las consultas médicas privadas también se beneficiarían de su uso, ya que podrían atender a más pacientes y por tanto mejorar sus servicios.

Sin embargo, aún se debe investigar en mayor profundidad en estos sistemas, de forma que se pueda obtener alguno cuyo coste de producción en masa sea asumible para la entidad que lo use, y que ofrezca un sistema de comunicación con una interfaz estándar interoperable entre distintos sistemas. Además, son deseables sistemas que proporcionen una batería cada vez más longeva, ya que su reemplazamiento muchas veces requiere acudir a un centro especializado y en algunos casos, operación ambulatoria.

Por otro lado, es necesario también tener en cuenta la privacidad de los datos recabados, ya que la información que se extrae en algunos casos puede puede ser sensible para el usuario: en países como por ejemplo EEUU que no poseen una sanidad pública, el que las aseguradoras puedan acceder a esta información puede resultar en que se deniegue la contratación de un seguro de salud. En este aspecto sería necesaria bajo mi punto de vista una legislación que regulara fuertemente la adquisición de los datos y el uso que se hace de ellos.

Otro aspecto importante es la confianza y el conocimiento que los usuarios poseen sobre los sistemas WBAN, ya que la información que se obtiene en los medios sobre avances en el campo no es mucha. Hay gente además que no se siente cómoda con tener un dispositivo electrónico implantado, gente que cree que sus datos no sólo van a ser utilizados para fines médicos, y gente que piensa que se van a recabar más datos aparte de los puramente médicos. Para solucionar este problema creo que la solución más adecuada es una campaña de información y sensibilización por parte de las entidades médicas mediante la cual el usuario pueda conseguir la información necesaria y se pueda cambiar la forma de ver estos sistemas.

Como conclusión respecto al desarrollo del Trabajo de Fin de Grado se puede decir que la plataforma eZ430-Chronos es una herramienta buena para hacer proyectos sencillos, pero acusa ciertas carencias en cuanto el proyecto se empieza a complicar; sería conveniente tener una memoria más grande que permitiera no tener que optimizar tanto el código, o un entorno de desarrollo que no se bloqueara de manera que fuera más fácil realizar el *debugging*.

Respecto del dispositivo en sí, puede tener interés utilizarlo como *hub* de una WBAN, ya que se liberaría parte del código que controla los sensores y únicamente actuaría como almacenamiento de los datos enviados desde los mismos. Además, tiene capacidad para proporcionar conectividad con dispositivos que sigan el estándar IEEE.

Referencias:

[1] Testing BAN (WBAN) Networks and IEEE 802.15.6, National Instruments, 2014
[2] Impact of Health IT on Nurses' Time Spent on Direct Patient Care, AHRQ Publication No: 090074 May 2009
[3] A Review of IEEE 802.15.6 MAC, PHY, and Security Specifications, Hindawi Publishing Corporation, 2013
[4] eZ430-ChronosTM Development Tool Users Guide, Texas Instruments, 2009
[5] eZ430-Chronos Watch, BOM, 868, 915, V1.5f, Texas Instruments
[6] cma3000-d0x_product_family_specification_8281000a.05, Murata
[7] COM-09478-scp1000_product_family_specification_rev_0.08, VTI Technologies
[8] cc430f6137, Texas Instruments, 2013
[9] eZ430-Chronos Watch, Schematics, 868, 915 V1.5, Texas Instruments
[10] eZ430-Chronos Access Point, BOM, 868 V1.3d, Texas Instruments
[11] Contempo: A Home Care Model to Enhance the Wellbeing of Elderly People, Kurnianingsih et al, 2014
[12] Long-Term Telemonitoring of Mobility Trends of Elderly People Using SMS Messaging, Cliodhna Ni Scanaill, Brian Ahearne, and Gerard M. Lyons, 2006
[13] Distributed intelligent architecture for falling detection and physical activity analysis in the elderly, M. Prado, J. Reina-Tosina, L. Roa, 2002
[14] A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound—Proof of Concept on Human Mimicking Doll Falls, Yaniv Zigel, Dima Litvak, and Israel Gannot, 2009
[15] SimpliciTI: Simple Modular RF Network Specification, Larry Friedman, 2009
[16] Regulations and Standards for Wireless Medical Applications, Salim A. Hanna
[17] BM-CS5 Chest Strap Datasheet Rev 1_0.pdf, BMinnovations, 2009
[18] BlueRobin White Paper 080812, BMinnovations, 2012
[19] Characterizing Tomorrow's Medical Devices, Robert Green, 2014

ANEXOS

Organización en carpetas

Observando en la estructura del código en C se comprueba que se organiza en varias carpetas, dentro de las cuales se encuentran diversos header files para el control de los periféricos del sistema, y los archivos en C a los que hacen referencia. Las carpetas y sus contenidos son los siguientes:

- Driver; contiene ficheros encargados de la inicialización de todo el hardware del sistema, así como la interacción con el mismo. Son ficheros en los que se configuran los puertos del MSP como puertos de entrada o de salida, y se asignan las interrupciones correspondientes. De igual manera se implantan protocolos de comunicación serie (SPI e I2C) para escribir o leer de registros de los periféricos.
- Logic; contiene ficheros con la lógica de cada módulo. Hacen uso de las funciones definidas en los ficheros de la carpeta Driver para interactuar con el hardware. Todas las funciones relacionadas con el uso de los módulos se encuentra aquí; funciones como por ejemplo hacer una lectura de la memoria de valores de calibración y escribirlos en el módulo correspondiente.
- SimpliciTI; dentro de ella están todas las librerías de uso del protocolo SimpliciTI para comunicación con el punto de acceso ya compiladas. También se encuentran algunos ficheros que se pueden consultar, en los que se listan los comandos de control del dispositivo disponibles.
- BlueRobin; al igual que en la carpeta SimpliciTI se encuentran librerías para la comunicación por radiofrecuencia con la banda de pecho mediante el protocolo BlueRobin, así como un header file para la consulta de funciones implementadas en las mismas.
- Include; carpeta con un único fichero en ella, project.h, que va incluido en todos los demás ficheros del sistema. En project.h se almacenan todas las variables globales en struct de forma que sean de más fácil acceso.

Ejemplo de un modo de funcionamiento

Es interesante comentar cómo funciona un modo, ya que todos funcionan de una manera similar. Como ejemplo, se seleccionará el modo DataLog, correspondiente a la línea 2.

En primer lugar, al seleccionar el modo con el botón NUM, se genera una interrupción, en cuyo tratamiento se indica que hay que hacer una actualización de display. Para ello se hace uso de la información correspondiente indicada en su función de display. Una vez que se ha producido este cambio el usuario pulsará el botón que activa la función principal del modo (activable con DOWN).
En dicha función se activa un flag que indica que el modo está activo, de forma que en la siguiente interrupción generada por el temporizador se activará un flag del tipo request, una petición en este caso para registrar los datos correspondientes al modo en que esté configurado. Así, con cada interrupción del timer se realiza un registro de los datos (dependiendo del intervalo de registro puede ser de cada interrupción a cada 10

interrupciones), ya que tras cada interrupción se sale del modo LPM3 y se salta de función en función dependiendo del diagrama de ejecución de main indicado arriba.
Al igual que para activarlo, se desactiva pulsando el botón DOWN, que hace que se cambie el flag del modo a apagado y por tanto no se procesen las peticiones.